Arquiteturas Alternativas

Apresentação

Com o passar do tempo e a evolução das arquiteturas dos computadores, foi necessário criar novos padrões e identificá-los. É notável que o primeiro contato que se tem com um computador é geralmente o computador pessoal. Mas não podemos esquecer que existem outras arquiteturas para computadores que fogem da abordagem clássica de Von Neumann. Portanto, é importante ter o conhecimento de outras plataformas com arquitetura de paralelismo, multiprocessamento e superescalares.

Nesta Unidade de Aprendizagem você estudará arquiteturas alternativas e suas respectivas características.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Reconhecer Arquiteturas RISC.
- Identificar as Arquiteturas Alternativas.
- Contrastar Arquitetura RISC e CISC.

Infográfico

Acompanhe o infográfico com o conteúdo desta Unidade de Aprendizagem.

Conteúdo do Livro

Com o avanço tecnológico juntamente com o uso específico de certas aplicações, paralelamente surgiram arquiteturas de processamento mais indicadas para alguns tipos específicos de uso. Essas arquiteturas estão presentes em computadores, smartphones, videogames, etc.

Acompanhe um trecho da obra Princípios Básicos de Arquitetura e Organização de Computadores. Inicie seus estudos em Arquiteturas alternativas.

Boa leitura.

Princípios Básicos de

Arquitetura *e* Organização *de* Computadores

Segunda Edição

Linda Null *e* Julia Lobur

bookman

N969p Null, Linda.

Princípios básicos de arquitetura e organização de computadores / Linda Null, Julia Lobur ; tradução: Maria Lúcia Blanck Lisbôa ; revisão técnica: Carlos Arthur Lang Lisbôa. – 2. ed. – Porto Alegre : Bookman, 2010.

822 p. : il. color. ; 25 cm.

ISBN 978-85-7780-737-6

1. Ciência da computação. 2. Arquitetura dos computadores. 3. Organização de computadores. I. Lobur, Julia. II. Título.

CDU 004.2

Catalogação na publicação: Ana Paula M. Magnus – CRB-10/Prov-009/10

Qualidade nunca é um acidente, é sempre o resultado de boa intenção, esforço sincero, rumo inteligente e execução habilidosa, o que representa a sábia escolha de muitas alternativas.

—William A. Foster

Parece que alcançamos os limites do que é possível alcançar com a tecnologia de computadores, embora se deva ter cuidado com tais afirmações, uma vez que tendem a soar tolas em 5 anos.

—John Von Neumann, 1949

CAPÍTULO 9

Arquiteturas Alternativas

9.1 INTRODUÇÃO

Nossos capítulos anteriores excursionaram pelos fundamentos da tecnologia de computação. As apresentações enfocaram de forma distinta sistemas uniprocessados do ponto de vista de um praticante de ciência da computação. Esperamos que você tenha adquirido uma compreensão das funções dos diferentes componentes de hardware e possa ver como cada um contribui para o desempenho global do sistema. Este entendimento é vital não só para a concepção de hardware, mas também para a implementação eficiente de algoritmos. A maioria das pessoas ganha familiaridade com o computador por intermédio de suas experiências com computadores pessoais e estações de trabalho. Isto deixou intocada uma área importante da arquitetura de computadores: a das arquiteturas alternativas. Portanto, o objetivo deste capítulo é apresentar a você um pouco das arquiteturas que transcendem a abordagem clássica de von Neumann.

Este capítulo discute máquinas RISC, arquiteturas que exploram paralelismo em nível de instrução, arquiteturas e multiprocessamento (com uma breve introdução ao processamento paralelo). Começamos com o famoso debate RISC *versus* CISC para dar a você uma ideia das diferenças entre essas duas ISAs e suas respectivas vantagens e desvantagens.

A seguir fornecemos uma taxonomia, por meio da qual as diferentes arquiteturas podem ser classificadas, com uma visão sobre o modo como as arquiteturas paralelas se enquadram na classificação. Em seguida, consideramos temas relevantes para arquiteturas paralelas em nível de instrução, enfatizando arquiteturas superescalares e reapresentando projetos EPIC (computadores com instruções explicitamente paralelas) e VLIW (palavra de instrução muito longa). Finalmente, apresentamos uma breve introdução aos sistemas multiprocessados e algumas abordagens alternativas de paralelismo.

Projetistas de hardware começaram a reavaliar vários princípios arquitetônicos no início dos anos 1980. O primeiro alvo dessa reavaliação foi a arquitetura do conjunto de instruções. Os projetistas se perguntaram por que razão uma máquina necessitava de um extenso conjunto de instruções complexas quando apenas cerca de 20%

das instruções eram usadas na maior parte do tempo. Esta questão levou ao desenvolvimento de máquinas RISC, que foram apresentadas pela primeira vez nos capítulos 4 e 5, e às quais vamos agora dedicar uma seção inteira do presente Capítulo. A onipresença de projetos RISC tem levado a um casamento ímpar de CISC com RISC. Muitas arquiteturas agora adotam núcleos RISC para implementar arquiteturas CISC.

Os Capítulos 4 e 5 descreveram como novas arquiteturas, tais como VLIW, EPIC e multiprocessadores, estão dominando uma grande percentagem do mercado de computadores. A invenção de arquiteturas que exploram o paralelismo em nível de instrução tem levado a técnicas que predizem precisamente o resultado de desvios no código do programa antes que o código seja executado. A carga antecipada de instruções baseada nestas predições tem aumentado grandemente a performance de computadores. Além de predizer a próxima instrução a ser carregada, altos graus de paralelismo em nível de instrução originaram ideias como execução especulativa, onde o processador tenta adivinhar o valor de um resultado antes que ele tenha sido realmente calculado.

O assunto de arquiteturas alternativas também inclui sistemas multiprocessados. Para estas arquiteturas, retornamos à lição que aprendemos de nossos ancestrais, a da amigável junta de bois. Se estamos usando uma junta de bois para puxar uma árvore e a árvore é muito grande, não tentamos fazer uma junta de bois maior. Em vez disso, usamos duas juntas de bois. Arquiteturas para multiprocessamento são análogas a juntas de bois. Precisamos delas se quisermos mover os troncos de problemas intratáveis. No entanto, os atuais sistemas multiprocessados nos apresentam desafios próprios, particularmente no que diz respeito à coerência de cache e à consistência de memória.

Notamos que, embora algumas destas arquiteturas alternativas estejam se consolidando, o seu verdadeiro avanço depende de seus custos incrementais. Atualmente, o relacionamento entre o desempenho exibido por sistemas avançados e seus custos é não linear, com o custo excedendo de longe os ganhos de performance na maioria das situações. Isto torna proibitivo o custo de integrar estas arquiteturas em aplicações atualmente em voga. Contudo, as arquiteturas alternativas têm seu lugar no mercado. Aplicações científicas e de engenharia altamente numéricas demandam máquinas que excedam a performance de sistemas uniprocessados. Para computadores deste tipo, o custo geralmente não é um problema.

Ao ler este capítulo, tenha em mente as gerações anteriores de computadores apresentadas no Capítulo 1. Muitas pessoas acreditam que estamos iniciando uma nova geração baseada nestas arquiteturas alternativas, em particular na área de processamento paralelo.

9.2 MÁQUINAS RISC

Introduzimos arquiteturas RISC no contexto de alguns sistemas exemplares no Capítulo 4. Relembre que máquinas RISC são assim denominadas porque elas originalmente ofereciam um conjunto menor de instruções se comparado ao das máquinas CISC. À medida que máquinas RISC estavam sendo desenvolvidas, o termo “reduzido” se tornou algo mal empregado, e hoje muito mais ainda. A ideia original era fornecer um conjunto mínimo de instruções que poderiam realizar todas as operações essenciais: movimento de dados, operações da UAL e desvios. Somente instruções explícitas `load e store` tinham permissão para acessar a memória.

Projetos de conjuntos complexos de instruções foram motivados pelo alto custo de memória. Ter mais complexidade empacotada em cada instrução significa que os programas poderiam ser menores e, portanto, ocupariam menos espaço. ISAs CISC adotam instruções de tamanho variável, mantendo curtas as instruções simples, ao mesmo tempo em que permitem instruções longas mais complicadas. Além disso, arquiteturas CISC incluem um grande número de instruções de acesso direto à memória. Assim, o que temos neste ponto é um conjunto de instruções de tamanho variável, denso e poderoso, que resulta em um número variável de ciclos de relógio por instrução. Algumas instruções complexas, particularmente aquelas instruções que acessam a memória, requerem centenas de ciclos. Sob certas circunstâncias, projetistas de computadores acharam necessário desacelerar o relógio do sistema (tornando maior o intervalo entre pulsos do relógio) a fim de dar tempo suficiente para as instruções se completarem. Tudo isso contribui para um tempo de execução mais longo.

Linguagens humanas exibem algumas das qualidades de RISC e CISC e servem como uma boa analogia para entender as diferenças entre as duas. Suponha que você tenha um amigo chinês. Suponha que vocês dois falam e escrevem fluentemente em inglês e chinês. Vocês dois desejam manter o custo de sua correspondência no mínimo, embora apreciem trocar longas cartas. Você deve escolher entre usar um papel de cartas bem caro, que economiza custos de correio, ou usar papel comum e pagar mais pelos selos. Uma terceira alternativa é compactar mais informação em cada página escrita.

Quando comparada à linguagem chinesa, a inglesa é simples, porém prolixa. Caracteres chineses são mais complexos do que palavras inglesas, e o que exigiria 200 letras em inglês poderia requerer somente 20 caracteres chineses. Corresponder-se em chinês requer menos símbolos, economizando papel e postagem. Entretanto, ler e escrever em chinês requer mais esforço porque cada símbolo contém mais informação. As palavras inglesas são análogas a instruções RISC, enquanto os símbolos chineses são análogos a instruções CISC. Para a maioria das pessoas que falam inglês, "processar" a letra em inglês pode levar menos tempo, mas também pode exigir mais recursos físicos.

Embora muitas fontes considerem RISC como um novo e revolucionário projeto, suas sementes foram lançadas na metade dos anos 1970, através do trabalho de John Cocke, da IBM. Cocke iniciou construindo seu mainframe experimental Modelo 801 em 1975. Esse sistema inicialmente recebeu pouca atenção, sendo seus detalhes revelados somente anos depois. Neste ínterim, David Patterson e David Ditzel publicaram seu amplamente aclamado "Case for a Reduced Instruction Set Computer", em 1980. Este artigo gerou uma maneira radicalmente nova de pensar sobre arquitetura de computadores e trouxe as siglas **CISC** e **RISC** para o léxico de ciência da computação. A nova arquitetura proposta por Patterson e Ditzel defendia instruções simples, todas do mesmo tamanho. Cada instrução realizaria menos trabalho, mas o tempo requerido para a execução da instrução poderia ser constante e previsível.

O suporte para máquinas RISC veio por intermédio de observações de programação em máquinas CISC. Estes estudos revelaram que as instruções de movimentação de dados contabilizavam aproximadamente 45% de todas as instruções. Operações da UAL (incluindo aritméticas, comparações e lógicas) contabilizavam 25%, e desvios (ou fluxo de controle) chegavam a 30%. Embora existissem muitas instruções complexas, poucas eram usadas. Esta constatação, aliada ao advento de memória mais barata e mais abundante e ao desenvolvimento da tecnologia VLSI levou a um tipo

diferente de arquitetura. Memória mais barata significava que programas podiam usar mais espaço de armazenamento. Programas maiores constituídos por instruções simples e previsíveis podiam substituir programas mais curtos formados por instruções mais complicadas de tamanho variável. Instruções simples permitiam o uso de ciclos de relógio mais curtos. Além disso, ter menos instruções poderia significar que menos transistores seriam necessários no chip. Menos transistores se traduzia em menor custo de fabricação e em mais espaço no chip disponível para outros usos. Instruções previsíveis acopladas com o avanço de VLSI poderiam permitir que diversos truques de aumento de performance, tais como pipelining, fossem implementados em hardware. CISC não fornece esta diversidade de oportunidades de melhoria de performance

Podemos quantificar as diferenças entre RISC e CISC usando a equação básica de performance de computador como segue:

$$\frac{\text{tempo}}{\text{programa}} = \frac{\text{tempo}}{\text{ciclo}} \times \frac{\text{ciclos}}{\text{instrução}} \times \frac{\text{instruções}}{\text{programa}}$$

A performance de um computador, quando medida pelo tempo de execução de um programa, é diretamente proporcional ao tempo de ciclo de relógio, ao número de ciclos de relógio por instrução e ao número de instruções do programa. Reduzir o ciclo de relógio, quando possível, resulta em melhoria de performance para RISC, bem como para CISC. Por outro lado, a máquina CISC aumenta a performance pela redução do número de instruções por programa. Computadores RISC minimizam o número de ciclos por instrução. Até agora ambas as arquiteturas podem produzir resultados idênticos em aproximadamente a mesma quantidade de tempo. No nível de portas, ambos os sistemas realizam uma quantidade equivalente de trabalho. Assim, o que está ocorrendo entre o nível de programa e o nível de porta?

Máquinas CISC se apoiam em microcódigo para lidar com a complexidade de instruções. O microcódigo diz ao processador como executar cada instrução. Por razões de desempenho, o microcódigo é compacto, eficiente e certamente deve estar correto. A eficiência do microcódigo, contudo, é limitada pelas instruções de tamanho variável, que tornam mais lento o processo de decodificação, e pelo número variável de ciclos de relógio por instrução, que torna mais difícil implementar pipelines de instruções. Além disso, o microcódigo interpreta cada instrução que é carregada da memória. Este processo de tradução adicional toma tempo. Quanto mais complexo o conjunto de instruções, mais tempo demora para examinar a instrução e acionar o hardware adequado para a sua execução.

Arquiteturas RISC usam uma abordagem diferente. A maior parte das instruções RISC são executadas em um ciclo de relógio. Para conseguir esta aceleração, o controle microprogramado é substituído por controle implementado em hardware, que é mais rápido na execução de instruções. Isto torna mais fácil fazer pipelining de instruções, mas mais difícil lidar com a complexidade em nível de hardware. Em sistemas RISC, a complexidade removida do conjunto de instruções é elevada em um nível, em direção ao domínio do compilador.

Para ilustrar, vamos examinar uma instrução. Suponha que queremos calcular um produto, 5×10. O código em uma máquina CISC poderia se parecer com isto:

```
mov ax, 10
mov bx, 5
mul bx, ax
```

Uma ISA RISC minimalista não possui instruções de multiplicação. Portanto, em um sistema RISC nossa multiplicação poderia se parecer com isto:

```
        mov ax, 0
        mov bx, 10
        mov cx, 5
Begin:  add ax, bx
        loop Begin   ;repete o laço cx vezes
```

O código CISC, embora mais curto, requer mais ciclos de relógio para executar. Suponha que, em cada arquitetura, operações de movimento registrador-a-registrador, soma e laço consumam um ciclo de relógio. Suponha também que uma operação de multiplicação requeira 30 ciclos de relógio.* Comparando os dois fragmentos de código temos:

Instruções CISC:

Total de ciclos de relógio = (2 `movs` × 1 ciclo de relógio) + (1 `mul` × 30 ciclos de relógio)
= 32 ciclos de relógio

Instruções RISC:

Total de ciclos de relógio = (3 `movs` × 1 ciclo de relógio) + (5 `adds` × 1 ciclo de relógio)
+ (5 `loops` × 1 ciclo de relógio)
= 13 ciclos de relógio

Adicione a isto o fato de que ciclos de relógio RISC são frequentemente mais curtos do que ciclos de relógio CISC, e deve estar claro que mesmo que existam mais instruções, o tempo real de execução é menor para RISC do que para CISC. Esta é a principal inspiração por trás do projeto RISC. Mencionamos que reduzir a complexidade de instruções resulta em chips mais simples. Os transistores antes usados na execução de instruções CISC são usados para pipelines, cache e registradores. Dentre os três, os registradores oferecem o maior potencial para melhoria de performance, de modo que faz sentido aumentar o número de registradores e usá-los de maneira inovadora. Uma destas inovações é o uso **conjunto de janelas de registradores**. Embora não seja amplamente aceito como outras inovações associadas com a arquitetura RISC, colocar registradores em janelas é, sem dúvida, uma ideia interessante e será rapidamente apresentada aqui.

Linguagens de alto nível dependem de modularização para eficiência. Chamadas de procedimentos e passagem de parâmetros são efeitos colaterais naturais do uso de módulos. Chamar um procedimento não é uma tarefa trivial. Envolve salvar um endereço de retorno, preservar os valores dos registradores, passar parâmetros (seja colocando-os em uma pilha ou usando registradores), desviar para a sub-rotina e executar a sub-rotina. Após o término da sub-rotina, as modificações nos valores dos parâmetros devem ser salvas e os valores anteriores dos registradores devem ser restaurados antes de retornar a execução ao programa que fez a chamada. Salvar registradores, passar parâmetros e restaurar registradores envolve esforços e recursos consideráveis. Com chips RISC tendo capacidade para centenas de registradores, a sequência de salvamento e restauração pode ser reduzida a uma simples troca de ambientes de registradores.

* Este não é um número fora da realidade – uma multiplicação em um Intel 8088 requer 133 ciclos de relógio para dois números de 16 bits.

Para entender completamente este conceito, tente imaginar todos os registradores como sendo divididos em conjuntos. Quando um programa estiver sendo executado em um ambiente, somente um determinado conjunto de registradores é visível. Se o programa muda para um ambiente diferente (digamos que um procedimento é chamado), o conjunto de registradores visível para o novo ambiente. Por exemplo, enquanto o programa principal está sendo executado, talvez ele veja apenas os registradores de 10 a 19. Valores típicos para arquiteturas RISC reais incluem 16 conjuntos de registradores (ou **janelas**) com 32 registradores cada um. A UCP fica restrita a operar com apenas uma janela em um dado momento. Portanto, da perspectiva do programador, existem somente 32 registradores disponíveis.

Janelas de registradores, por si só, não necessariamente ajudam chamadas de procedimentos ou passagem de parâmetros. Entretanto, se estas janelas forem cuidadosamente sobrepostas, o ato de passar parâmetros de um módulo para outro se torna uma simples questão de mudar de um conjunto de registradores para outro, permitindo que dois conjuntos se sobreponham exatamente naqueles registradores que devem ser compartilhados para realizar a passagem de parâmetros. Isto é feito dividindo o conjunto de janelas de registradores em partições distintas, incluindo **registradores globais** (comuns a todas as janelas), **registradores locais** (locais para a janela corrente), **registradores de entrada** (que se sobrepõem aos registradores de saída da janela precedente) e **registradores de saída** (que se sobrepõem aos registradores de entrada da próxima janela). Quando a UCP muda de um procedimento para o seguinte, ela muda para uma janela diferente de registradores, mas as janelas sobrepostas permitem que parâmetros sejam "passados" simplesmente mudando dos registradores de saída do módulo que fez a chamada para os registradores de entrada do módulo que foi chamado. Um **apontador de janela corrente (CWP – *current window pointer*)** aponta para o conjunto de janelas de registradores a ser usado em um dado momento.

Considere um cenário no qual o `Procedimento Um` está chamado o `Procedimento Dois`. Dos 32 registradores em cada conjunto, assuma que 8 são globais, 8 são locais, 8 são de entrada e 8 são de saída. Quando o `Procedimento Um` chama o `Procedimento Dois`, quaisquer parâmetros que necessitem ser passados são colocados no conjunto de registradores de saída do `Procedimento Um`. Quando o `Procedimento Dois` inicia a execução, estes registradores se tornam o conjunto de registradores de entrada para o `Procedimento Dois`. Este processo é ilustrado na Figura 9.1.

Mais uma informação importante a notar em relação a janelas de registradores em máquinas RISC é a natureza circular do conjunto de registradores. Em programas que contenham um alto grau de aninhamento, é possível gastar o suprimento de registradores. Quando isto acontece, a memória principal atua, armazenando as janelas de números mais baixos, as quais contém valores das ativações mais antigas de procedimentos. As posições dos registradores de número mais alto (as ativações mais recentes) são, então, colocadas nos registradores de número mais baixo. À medida que são executados os retornos dos procedimentos, o nível de aninhamento diminui e os valores dos registradores na memória são restaurados na ordem na qual eles foram salvos.

Além de instruções simples de tamanho fixo, pipelines eficientes em máquinas RISC foram providenciados para estas arquiteturas com um enorme aumento de velocidade. Instruções mais simples liberaram espaço no chip, resultando não apenas em

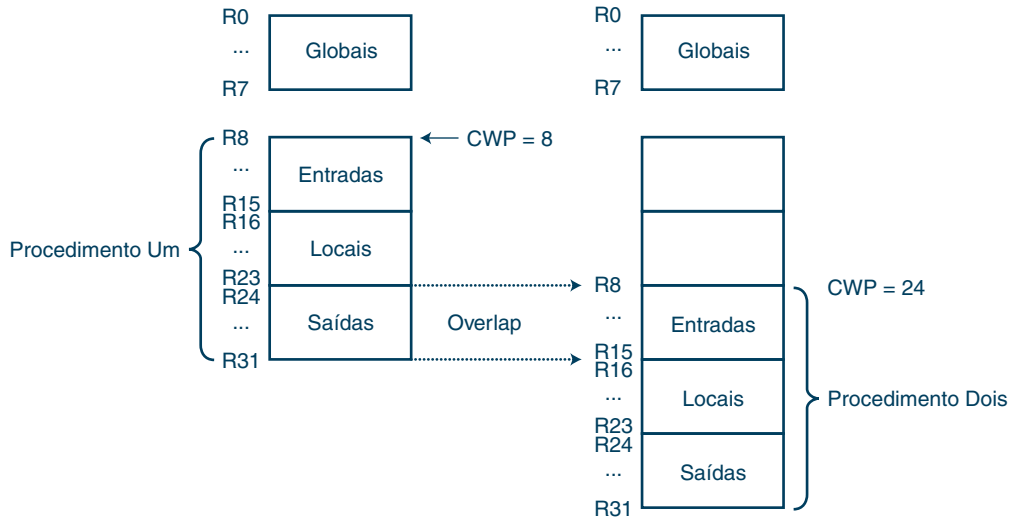

FIGURA 9.1 Sobrepondo janelas de registradores.

espaço mais usável, mas também em chips que são mais fáceis e menos demorados para projetar e fabricar.

Você deve estar ciente de que está se tornando cada vez mais difícil categorizar os processadores atuais como sendo RISC ou CISC. As linhas divisórias destas arquiteturas não são mais nítidas. Algumas arquiteturas correntes usam ambas abordagens. Se você olhar alguns dos novos manuais de chips, verá que as atuais máquinas RISC tem instruções mais extravagantes e complexas do que algumas máquinas CISC. O RISC PowerPC, por exemplo, tem um conjunto de instruções maior do que o Pentium CISC. À medida que a tecnologia VLSI continua a fazer transistores menores e mais baratos, a expansão do conjunto de instruções está agora se tornando uma questão menor do debate CISC *versus* RISC, enquanto o uso de registradores e de arquitetura carrega/armazena está se tornando mais proeminente.

Dito isso, cautelosamente fornecemos a Tabela 9.1 como um resumo das diferenças clássicas entre RISC e CISC.

Como mencionamos, embora muitas fontes exaltem as inovações revolucionárias do projeto RISC, muitas das ideias usadas em máquinas RISC (incluindo pipelining e instruções simples) foram implementadas em computadores de grande porte nos anos 1960 e 1970. Existem muitos dos assim chamados novos projetos que não são realmente novos, mas simplesmente reciclados. A inovação não necessariamente significa inventar uma nova roda; pode ser um simples caso de descobrir um modo melhor de usar uma roda que já existe. Esta é uma lição que servirá para você em sua carreira no campo de computação.

9.3 TAXONOMIA DE FLYNN

Ao longo dos anos, várias tentativas foram feitas a fim de encontrar uma maneira satisfatória para categorizar arquiteturas de computadores. Embora nenhuma delas

TABELA 9.1 As características de máquinas RISC *versus* máquinas CISC

RISC	CISC
Vários conjuntos de registradores, muitas vezes contendo mais de 256 registradores	Um conjunto de registradores, geralmente 6 a 16 registradores no total
Permite três operandos em registradores por instrução (p. ex., `add R1, R2, R3`)	Permite um ou dois operandos em registradores por instrução (p. ex., `add R1, R2`)
Passagem de parâmetros eficiente através de janelas de registradores no chip	Passagem de parâmetros ineficiente em memória fora do chip
Instruções de um ciclo (exceto para `load` e `store`)	Instruções de vários ciclos
Controle implementado em hardware	Controle microprogramado
Muito pipeline	Pouco pipeline
Pequeno número de instruções simples	Muitas instruções complexas
Instruções de tamanho fixo	Instruções de tamanho variável
Complexidade no compilador	Complexidade no microcódigo
Apenas instruções de carga e armazenamento podem acessar a memória	Muitas instruções podem acessar a memória
Poucos modos de endereçamento	Muitos modos de endereçamento

seja perfeita, a taxonomia mais amplamente aceita atualmente é aquela proposta por Michael Flynn em 1972. A **Taxonomia de Flynn** considera dois fatores: o número de instruções e o número de sequências de dados que passam para o processador. Uma máquina pode ter uma ou várias sequências de dados e pode ter um ou vários processadores trabalhando nestes dados. Isto nos dá quatro possíveis combinações: **SISD** (***single instruction stream***, ***single data stream***), **SIMD** (***single instruction stream***, ***multiple data streams***), **MISD** (***multiple instruction streams***, ***single data stream***) e **MIMD** (***multiple instruction streams***, ***multiple data streams***).

Uniprocessadores são máquinas SISD. Máquinas SIMD, que possuem um único ponto de controle, executam a mesma instrução simultaneamente sobre diferentes valores de dados. A categoria SIMD inclui processadores de arrays, processadores de vetores e arrays sistólicos. Máquinas MISD possuem várias sequências de instruções operando sobre a mesma sequência de dados. Máquinas MIMD, que adotam vários pontos de controle, possuem sequências de instruções e dados independentes. Multiprocessadores e a maioria dos sistemas paralelos atuais são máquinas MIMD. Computadores SIMD são mais simples de projetar do que máquinas MIMD, mas também são consideravelmente menos flexíveis. Todos os multiprocessadores SIMD devem executar a mesma instrução simultaneamente. Se você pensar a respeito, executar algo tão simples como um desvio condicional pode rapidamente se tornar muito caro.

A taxonomia de Flynn deixa a desejar em muitas áreas. Primeiramente, parece que existem muito poucas aplicações (se houver) para máquinas MISD. Segundo, Flynn assumiu que o paralelismo era homogêneo. Uma coleção de processadores pode ser homogênea ou heterogênea. Uma máquina pode conceitualmente ter quatro somadores de ponto-flutuante separados, dois multiplicadores e uma única unidade

aritmética para inteiros. Esta máquina pode, portanto, executar sete operações em paralelo, mas não cabe facilmente no sistema de classificação de Flynn.

Outro problema com esta taxonomia é com a categoria MIMD. Uma arquitetura com diversos processadores recai nesta categoria sem considerar como os processadores são conectados ou como eles enxergam a memória. Existem várias tentativas de refinar a categoria MIMD. Alterações sugeridas incluem subdividir MIMD para diferenciar sistemas que compartilham memória daqueles que não, bem como categorizar processadores conforme eles sejam baseados em barramento ou chaveados.

Sistemas de memória compartilhada são aqueles nos quais todos os processadores têm acesso a uma memória global e se comunicam por variáveis compartilhadas, da mesma forma que processos em uniprocessadores. Se vários processadores não compartilham memória, cada processador deve possuir uma porção de memória. Consequentemente, todos os processadores devem se comunicar por meio de passagem de mensagens, o que pode ser caro e ineficiente. O problema que as pessoas têm com usar memória como um fator determinante de classificação de hardware é que memória compartilhada e passagem de mensagens são, na verdade, modelos de programação, e não, modelos de hardware. Portanto, pertencem mais propriamente ao domínio de software de sistema.

Os dois maiores paradigmas de arquitetura paralela, **multiprocessadores simétricos** (**SMPs**) e **processadores massivamente paralelos** (**MPPs**), são ambos arquiteturas MIMD, mas diferem na maneira como usam a memória. Máquinas SMP, tais como o processador dual Intel PC e o Silicon Graphics Origin 3900 (que suporta até 512 processadores), compartilham memória, enquanto processadores MPP, tais como nCube, CM5 e Cray T3E, não. Estas máquinas MPP, em particular, geralmente contêm milhares de UCPs em um único gabinete grande conectado a centenas de gigabytes de memória. O preço destes sistemas pode atingir milhões de dólares.

Originalmente, o termo MPP descrevia multiprocessadores SIMD fortemente acoplados, tais como o Connection Machine e o MPP da Goodyear. Hoje, entretanto, o termo MPP é usado para referir-se a arquiteturas paralelas que possuem diversos nodos autocontidos com memórias privativas, todos com a habilidade de se comunicar via redes. Uma maneira fácil de diferenciar SMP e MPP (pelas definições atuais) é a seguinte;

> MPP = muitos processadores + memória distribuída + comunicação via rede

e

> SMP = poucos processadores + memória compartilhada + comunicação via memória

Computadores MPP são difíceis de programar porque o programador deve assegurar que as partes do programa que estão sendo executadas em UCPs separadas possam se comunicar entre si. Contudo, máquinas SMP apresentam sérios gargalos quando todos os processadores tentam acessar a mesma memória ao mesmo tempo. A decisão de usar MPP ou SMP depende da aplicação – se o problema é facilmente particionável, MPP é uma boa opção. Grandes empresas frequentemente usam sistemas MPP para armazenar dados de clientes (depósito de dados) e para realizar mineração de dados sobre esses dados.

A computação distribuída é um outro exemplo de arquitetura MIMD. A **computação distribuída** (coberta com mais detalhe na Seção 9.4.5) é geralmente definida como um conjunto se computadores em rede que trabalham em colaboração para resolver um problema. Esta colaboração, entretanto, pode ocorrer de muitas maneiras diferentes.

Uma **rede de estações de trabalho (NOW – *network of workstations*)** é uma coleção de estações de trabalho distribuídas que trabalham em paralelo somente enquanto os nodos não estão sendo usados como estações de trabalho comuns. NOWs geralmente consistem de sistemas heterogêneos, com diferentes processadores e software, que se comunicam via Internet. Usuários individuais devem estabelecer a conexão apropriada na rede antes de se juntar à computação paralela. NOWs são frequentemente usadas dentro de organizações que possuem intranets, nas quais todas as estações de trabalho podem ser controladas. Um **agrupamento de estações de trabalho (COW – *cluster of workstations*)** é uma coleção similar a uma NOW, mas requer que uma única entidade seja responsável. Os nodos geralmente usam um mesmo software, e um usuário que pode acessar um nodo pode geralmente acessar todos os nodos. Um agrupamento dedicado de computadores paralelos **(DCPC – *dedicated cluster parallel computer*)** é uma coleção de estações de trabalho reunidas especificamente para trabalhar numa dada computação paralela. As estações de trabalho têm os mesmos software e sistema de arquivos, são gerenciadas por uma única entidade, se comunicam via Internet e não são usadas como estações de trabalho. Uma **pilha de PCs (PoPC – *pile of PCs*)** é um agrupamento heterogêneo de hardware dedicado usado para construir um sistema paralelo a partir de componentes do mercado, ou COTs. Enquanto um DCPC tem relativamente poucos componentes, porém caros e rápidos, um PoPC usa um grande número de nodos lentos, porém relativamente baratos. NOWs, COWs, DCPCs e PoPCs são exemplos de **computação agrupada**, computação distribuída nas quais os recursos estão todos dentro do mesmo domínio administrativo, trabalhando em tarefas de grupo.

O projeto BEOWULF, apresentado em 1994 por Thomas Sterling e Donald Becker, do Goddard Space Flight Center, é uma arquitetura PoPC que reuniu com sucesso diversas plataformas de hardware com software especialmente projetado, resultando em uma arquitetura que tem a aparência e o comportamento de uma máquina paralela unificada. Um agrupamento BEOWULF tem três características definidoras: computadores pessoais de prateleira, rápido chaveamento de dados e software livre. Os nodos numa rede BEOWULF são sempre conectados via Ethernet privada ou rede ótica. Se você tem uma Sun SPARC antiga, um par de máquinas 486, uma DEC Alpha (ou simplesmente uma grande coleção de máquinas Intel empoeiradas) e um meio de conectá-las em uma rede, você pode instalar o software BEOWULF e criar o seu próprio computador paralelo, extremamente poderoso.

A taxonomia de Flynn foi recentemente expandida para incluir arquiteturas **SPMD (*single program multiple data*)**. Uma SPMD consiste de multiprocessadores, cada um com seu conjunto de dados e memória de programa. O mesmo programa é executado em cada processador, com sincronização em diversos pontos de controle global. Embora cada processador carregue o mesmo programa, cada um pode executar diferentes instruções.

Por exemplo, um programa pode ter um código que se parece com:

```
If myNodeNum = 1 do this, else do that
```

Desta maneira, nodos diferentes executam instruções diferentes dentro do mesmo programa. SPMD é atualmente um paradigma de programação usado em máquinas MIMD e difere de SIMD no sentido de que os processadores podem fazer

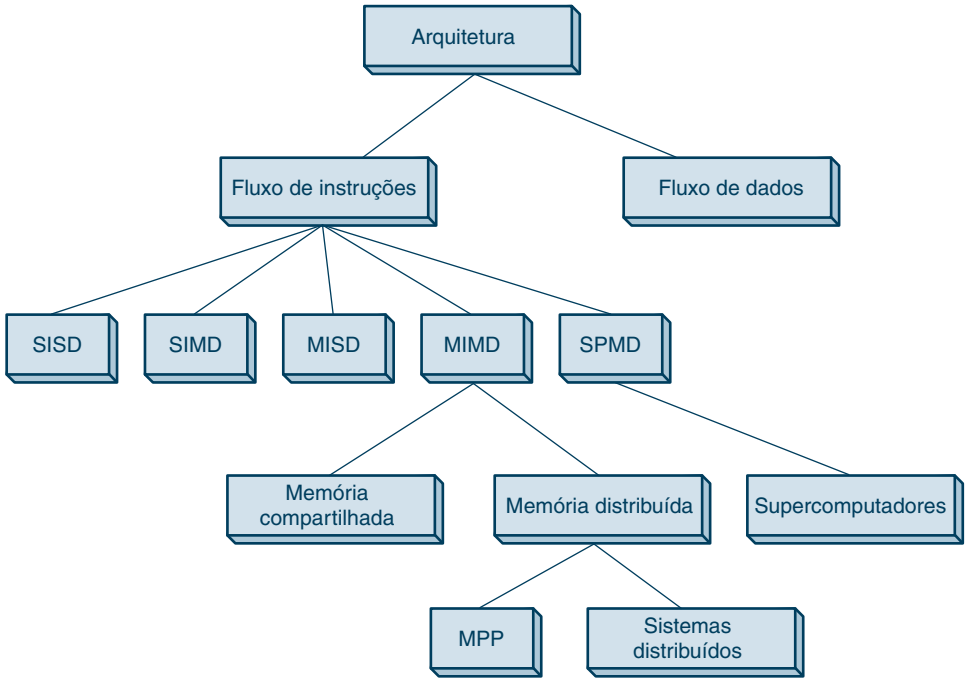

FIGURA 9.2 Uma taxonomia de arquiteturas de computadores.

coisas diferentes ao mesmo tempo. Supercomputadores frequentemente usam um projeto SPMD.

Em um nível acima daquele onde Flynn inicia a sua taxonomia, precisamos adicionar mais uma característica que indique se a arquitetura é orientada a instruções ou orientada a dados. Todas as atividades do processador são determinadas por uma sequência de código de programa. As instruções do programa atuam sobre os dados. Arquiteturas orientadas a dados, ou ***dataflow***, fazem justamente o oposto. As características dos dados determinam a sequência de eventos do processador. Vamos explorar esta ideia com mais detalhes na Seção 9.5.

Com a adição de computadores de fluxo de dados e alguns refinamentos na classificação MIMD, obtemos a taxonomia mostrada na Figura 9.2. Você pode requer consultá-la quando ler as seções seguintes. Iniciamos com o ramo esquerdo da árvore, com tópicos relevantes para arquiteturas SIMD e MIMD.

9.4 ARQUITETURAS PARALELAS E MULTIPROCESSADAS

Desde o início da computação, os cientistas têm se empenhado para fazer as máquinas resolverem problemas de forma melhor e mais rápida. A tecnologia da miniaturização resultou em circuitos melhores e em mais deles em um chip. Os relógios se tornaram mais rápidos, levando a UCPs na faixa de gigahertz. Entretanto, sabemos que existem barreiras físicas que controlam a extensão na qual a performance de uniprocessadores pode ser melhorada.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Dica do Professor

No vídeo desta unidade você vai aprofundar os seus conhecimentos sobre arquiteturas alternativas.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **Em relação à arquitetura RISC, qual afirmação está CORRETA?**

A) a) Apresenta poucos registradores.

B) b) Consome vários pulsos de Clock de processamento.

C) c) Utiliza muito pipeline.

D) d) Instruções RISC são executadas pelo microcódigo.

E) e) Tem pelo menos quatro fases de processamento de instruções.

2) **Com relação à arquitetura RISC, é CORRETO afirmar:**

A) a) As informações concebidas pelos compiladores são armazenadas na RAM.

B) b) As informações concebidas pelos compiladores são armazenadas na ROM.

C) c) As informações concebidas pelos compiladores são armazenadas na CPU.

D) d) As informações concebidas pelos compiladores são armazenadas na nuvem.

E) e) As informações concebidas pelos compiladores são armazenadas na EPROM.

3) **NÃO são características de um processador CISC:**

A) a) Arquitetura Registrador-Memória.

B) b) Muita variedade de dados.

C) c) Instruções com muitos endereços.

D) d) Acesso a dados via registradores.

E) e) 20 e 30 estágios de pipeline.

4) **Qual a definição de CISC?**

A) a) Computador com um conjunto complexo de instruções.

B) b) Arquitetura sem um conjunto de instruções complexas.

C) c) Computador com um conjunto de instruções simples.

D) d) Computador com um conjunto reduzido de instruções.

E) e) Computador sem conjunto de instruções.

5) **Entre os processadores abaixo, qual NÃO é RISC?**

A) a) ARM.

B) b) PowerPC.

C) c) XAP.

D) d) Super Hitachi.

E) e) 486.

Na prática

Fabricantes de videogames, por exemplo, costumam ter departamentos internos de desenvolvimento de hardware e software de jogos. Isso permite que os produtos sejam únicos e tenham preços competitivos frente ao mercado.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Como funciona a arquitetura de um processador Intel Pentium

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Arquitetura RISC

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Arquitetura de processadores: RISC e CISC

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

810: do que o super chip da Qualcomm é capaz e o que você ganha?

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.