Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação

Disciplina SCC0510
Arquitetura de Computadores

Grupo nr.: 13
Nome: Guilherme Marchetto NUSP: 9897988

1ª Questão: Explique o que são, compare e exemplifique as arquiteturas SISD, SIMD, MISD, MIMD.

Resposta: Essas siglas referem-se à Classificação de FLynn, a mais conhecida para classificar a arquiteturas das máquinas quanto a unicidade de multiplicidade dos fluxos de dados e de instruções. A seguir um esquema que representa essa classificação:

		Fluxo de Dados	
		Único	Múltiplo
Fluxo de Instruções	Único	SISD	SIMD
	Múltiplo	MISD	MIMD

Cada uma tem suas particularidades e aplicações:

- SISD (Single Instruction, Single Data): trata-se das arquiteturas puramente sequenciais, como na arquitetura de Von Neumann convencional. Um único fluxo de instruções é aplicado sobre um único fluxo de dados. Isso não significa que não se pode haver pipeline ou várias threads, nesse caso cada instrução é organizada na ordem que for mais conveniente.
- SIMD (Single Instruction, Multiple Data): representa os processadores vetoriais, matriciais, paralelos e associativos. Uma única unidade de controle envia a mesma instrução para vários processadores que a executam sobre múltiplos operandos ao mesmo tempo.

  Máquinas como essa são produzidas pela Cambridge Parallel Processing Inc. (CPP) que disponibiliza modelos com matrizes de 32x32 e 64x64 processadores, normalmente conectado a um computador front-end por uma

interface para execução em paralelo. São úteis para processamento de imagens, cálculos aritméticos com variáveis com grande número de bits e pontos flutuantes, bancos de dados, modelos de engenharia e processamentos de sinais.

- MISD (Multiple Instruction, Single Data): cada unidade de processamento executa uma instrução sobre o dado e passa o resultado à seguinte, que executará uma instrução diferente, ou seja, várias unidades de processamento executam instruções distintas sobre o mesmo fluxo de dados. Como exemplo temos os filtros de frequência e efeitos aplicados pelos pedais e alavancas sobre o sinal de uma guitarra elétrica ou máquinas usadas para tarefas específicas de criptografia e descriptografia.
- MIMD (Multiple Instruction, Multiple Data): vários processadores recebem instruções diferentes e operam, de forma síncrona ou assíncrona, sobre fluxo de dados diferentes. Nesse caso faz-se necessária a presença de uma unidade de controle independente para cada unidade de processamento. Essa classificação representa os multiprocessadores

2ª Questão: Explique o que são, compare e exemplifique as arquiteturas com memória compartilhada e memória distribuída.

Resposta: Multiprocessadores, ou máquinas de memória compartilhada, são aquelas em que vários processadores executam instruções em paralelo e fazem leitura e escrita de dados na mesma memória. Toda a sincronização e coerência entre as threads é feita por esse acesso compartilhado à memória, pois os diferentes processadores terão acesso aos mesmos dados do programa e um de cada vez.

Podem ser de multiprocessamento simétrico, formato no qual todos os processadores têm o mesmo privilégio de acesso e as tarefas são executadas em qualquer um que esteja disponível, como é o caso das SGI Power Challenge e máquinas da Sun Microsystems e da Silicon Graphics. Outro formato é o de acesso não-uniforme à memória, em que cada processador tem seus bancos de memória locais e o conjunto de todos é a memória principal da máquina, ou seja, caso necessário um processador deve solicitar o acesso ao banco de memória do outro. A latência de acesso é diferente para cada segmento da memória, por isso é chamado de “não-uniforme”. É o caso das arquiteturas do 1024 MIPS R10000 e 252

Pentium II da Silicon Graphics.

Nas arquiteturas de memória distribuída cada processador tem uma memória exclusiva para si, fisicamente impossível de ser acessada por outro. A sincronização e compartilhamento de dados entre os processadores são feitos através da comunicação por mensagens, por esse motivo são difíceis de programar.

Para máquinas com processadores massivamente paralelos cada um é interconectado por uma rede de alta velocidade que facilita a comunicação entre eles e a escalabilidade do sistema, como o Intel Paragon, Cray T3E e o Thinking Machines CM-5. No formato de clusters diferentes estações de trabalho são conectadas através de uma rede local que trabalham como um único recurso computacional, o escalonamento é feito adicionando novas máquinas à rede. Cada máquina executa uma parte da tarefa e o tratamento de erros, redundâncias e coerência dos dados é feito por um software.

3ª Questão: Explique o que são, compare e exemplifique as seguintes máquinas:

1. **Processador com pipeline de operações**
2. **Processadores Superescalares**
3. **Processadores Paralelos**

Resposta: Para um processador executar um programa ele o divide em vários ciclos de operação, como os de buscar instrução, decodificar, executar a operação, entre outros. Cada um dos ciclos utiliza barramentos e recursos diferentes do processador. Um processador com pipeline organiza os ciclos de diferentes instruções em série para serem executados no mesmo ciclo de clock, agilizando o processamento. O recurso de decodificar a instrução, o de executar uma operação aritmética e o acesso à memória são independentes, portanto é possível executar cada uma dessas partes de instruções diferentes que seriam executadas em série, por exemplo. Apesar dos grandes ganhos em desempenho é necessário fazer um tratamento de interrupções, como abordado na questão 6, e de concorrência de recursos. A técnica de pipeline foi utilizada desde os processadores 485 da Intel, que usavam um pipeline de 5 etapas.

Processadores superescalares possuem diversos pipelines, neles se busca as próximas instruções independentes às que estão sendo executadas para executá-las em paralelo, levando-se se em conta possíveis dependências e

conflitos. Em 1990, a IBM anunciou o primeiro processador superescalar de arquitetura RISC, o IBM Power-1 da linha RS/6000.

Para processadores paralelos unidades de processamento independentes, com barramentos e recursos diferentes executam as instruções de um programa ao mesmo tempo. O programa a ser executado pode ser dividido em tarefas menores que então são resolvidas concorrentemente, como é o caso de processadores multinúcleos.

4ª Questão: Explique as limitações intrínsecas do paralelismo: Dependência de dados, Dependência de desvio, Conflito de recurso (ULA) e o que pode ser feito para minimizar esses problemas.

Resposta: Quando a execução de uma instrução depende do término da execução de outra temos o que é chamado de dependência de dados. Por exemplo, quando é necessário armazenar o resultado de uma operação aritmética na memória, primeiro termina-se a execução da operação, depois executa-se a instrução de armazenamento. É um tipo de dependência muito comum e para resolvê-la o compilador pode reorganizar a ordem das instruções para que não sejam executadas uma seguida da outra, assim enquanto a primeira está no pipeline outras instruções independentes são postas em sequência e só depois a instrução com dependência entra no pipeline.

A dependência de desvio pode ser causada pelo próprio programa ou por uma interrupção. As instruções a serem executadas depois do desvio vão depender do resultado de sua execução, vão sendo carregadas mais instruções no pipeline e se ocorrer o desvio estas são descartadas e o pipeline é carregado com as novas instruções. Algumas técnicas para minimizar esse problema são o delayed branch e otimização de branch, que serão abordadas na questão 6.

Quando duas ou mais instruções competem pelo mesmo recurso no processador, como a ULA, temos o que é chamado de conflito de recurso, que apresenta um comportamento semelhante ao da dependência de dados. Esses conflitos podem ser superados pela duplicação do recurso em questão ou com a implementação de um pipeline dedicado a este recurso.

5ª Questão: Explique renomeação de registradores.

Resposta: A renomeação de registradores é uma técnica utilizada em computação

paralela cujo objetivo é evitar uma interrupção quando há conflitos na utilização de um registrador.

Por exemplo, threads diferentes estão programadas para utilizar o mesmo registrador porém uma não afeta no resultado da outra nem a ordem de execução importa, sem essa técnica uma teria que esperar o término da outra para começar a executar, afinal estão concorrendo pelo uso do mesmo registrador, porém se há outros registradores ociosos a máquina pode realocar aquelas instruções para utilizar outro registrador e iniciar imediatamente sua execução. O registrador nomeado no programa foi renomeado pela máquina para agilizar o processamento, por isso o nome. Logicamente deve-se tratar de conflitos de dados e execuções posteriores devem levar em conta a renomeação feita.

6ª Questão: Explique o que são, compare e exemplifique as seguintes técnicas: Delayed Branch e Otimização do Branch

Resposta: A técnica de delayed branch consiste em o compilador encher o pipeline depois da instrução de desvio com NOOPs, instruções que não executam ação alguma, impedindo assim que operações indevidas sejam feitas sobre os dados antes que o desvio seja executado. A quantidade depende de quantos ciclos de operação o pipeline é dividido. O desvio é reendereçado levando-se em conta a quantidade de NOOPs adicionados.

Já na técnica de otimização de branch o compilador antecipa a instrução de desvio, dessa forma, ao invés de NOOPs, são executadas as instruções que já deveriam ser executadas antes do desvio, não havendo necessidade de esperar a execução do desvio para continuar a preencher o pipeline de instruções e os recursos computacionais não ficam ociosos enquanto se executa NOOPs. A quantidade de instruções passadas para depois do desvio também depende da quantidade de ciclos de operação do pipeline.