UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
Departamento de Ciências da Computação
Disciplina de Arquitetura de Computadores

Prova - Arquitetura de Computadores - GRUPO 11

Ana Carolina Simões Ramos - 11816035
Rafael Belisario de Oliveira - 10431452
Kalilo Teixeira Gonçalves - 11836754
Thiago Rodrigues da Cunha Marafeli - 11816101

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

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

3ª Questão: Explique o que são, compare e exemplifique as seguintes máquinas:
1. Processador com pipeline de operações
2. Processadores Superescalares
3. Processadores Paralelos

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.

5ª Questão: Explique renomeação de registradores.

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

1.

Essas 4 arquiteturas fazem parte da classificação de arquiteturas proposta por Flynn

- **SISD** significa Single Instruction stream, Single Data stream, ou seja, único fluxo de instrução e único fluxo de dados, de modo que apenas uma operação é feita por vez. As instruções são executadas serialmente, porém os estágios (busca da instrução, decodificação, busca do operando e execução) podem ser sobrepostos (pipeline). Um exemplo de SISD é o *Arduino*.

- **MISD** significa Multiple Instruction stream, Single Data stream, ou seja, múltiplo fluxo de instruções e único fluxo de dados. Isso quer dizer que vários processadores recebem instruções distintas mas operam sobre o mesmo conjunto de dados. Um exemplo de MISD são *vários pedais de guitarra conectadas, cada uma aplicando um filtro diferente*.

- **SIMD** significa Single Instruction stream, Multiple Data stream, ou seja, são caracterizadas por possuírem apenas uma unidade de controle que executa uma instrução de cada vez, mas sobre vários dados ao mesmo tempo. Existem duas categorias de arquitetura SIMD: processadores matriciais e vetoriais. Os processadores vetoriais servem para uso geral, pois realizam todas as operações escalares normais. Já os processadores matriciais geralmente são configurados como periféricos de outras arquiteturas para a execução de instruções vetoriais dos programas. Um exemplo de SIMD é uma *placa de vídeo*.

- Por fim, **MIMD** significa Multiple Instruction stream, Multiple Data stream, ou seja, múltiplo fluxo de instruções e múltiplo fluxo de dados. Isso quer dizer que, vários processadores recebem instruções diferentes e operam sob um fluxo de dados diferentes, podendo ser síncronos ou assíncronos. Um exemplo disso são os *computadores modernos multi-core*.

2.

- **Arquitetura com memória compartilhada**: Cada processador consegue ter acesso a todo o espaço de memória do sistema. Não há necessidade de se movimentar fisicamente os dados quando dois ou mais processadores se comunicam. Como resultado, a comunicação entre processos é bastante eficiente. Há necessidade do uso de técnicas de sincronização quando há o acesso a regiões compartilhadas na memória, para assegurar um resultado correto para a computação, e também há uma limitação de escalabilidade devido ao problema de contenção de memória, já que depois de um determinado número de processadores a adição de mais processadores não aumenta o desempenho. Exemplos: *Multiprocessadores simétricos ou SMP (Symmetric MultiProcessors) e arquitetura NUMA (Non-Uniform Memory Access).*

- **Arquitetura com memória distribuída**: Cada processador é capaz de endereçar apenas a sua memória local. É uma arquitetura altamente escalável e permite a construção de processadores paralelos, porém a comunicação entre eles se dá através de troca de mensagens. Essa troca de mensagens resolve tanto o problema da comunicação processadores como o da sincronização. Exemplos: *COW (Cluster of Workstations) e MPP (Massively Parallel Processors).*

3.

- **Processador com pipeline de operações**: Esse tipo de processador utiliza a técnica de pipeline, que é uma técnica de implementação na qual várias instruções são sobrepostas ao longo da execução. Elas são colocadas de forma que as instruções não precisem ser completamente finalizadas antes de iniciar a busca pela próxima instrução, fazendo assim com que se reduza em bastante tempo a execução das instruções. Exemplos: *Pentium II, Athlon Thunderbird e Pentium 4.*

- **Processadores superescalares**: Esses processadores possuem vários pipelines independentes e podem ser separados por recursos, agrupando instruções que utilizarão aquele mesmo recurso e as executando em paralelo,em um mesmo ciclo de clock. Elas são feitas para operar em múltiplas threads e podem executar

algumas instruções fora da ordem original. Isso permite otimizar o uso dos recursos e acelera a velocidade de processamento. Exemplos: *Intel Pentium Pro e Pentium II, AMD K, Sun UltraSPARC, Pentium III, AMD Athlon, Alpha 21264, entre outros.*

- **Processadores paralelos**: um conjunto de processadores que pode ser chamado também de computador paralelo síncrono com múltiplas unidades lógicas e aritméticas que operam em paralelo. Essas unidades são chamadas de elementos de processamento, as quais são sincronizadas com o intuito de fazer a mesma operação simultaneamente, compostos por uma ULA e uma memória local conectados por rede. A Unidade de Controle faz a busca e decodificação de instruções, controlando as interconexões entre os elementos e enviando as instruções vetoriais a eles para serem executadas sobre os operandos obtidos das memórias locais. Exemplos: *IBM S/390 Mainframe, Silicon Graphics, máquinas da Sun Microsystems*

O conceito de processadores paralelos está ligado ao objetivo geral de se obter um melhor desempenho por meio da técnica de paralelismo utilizando arquiteturas avançadas como os pipelines, o arranjo de processadores e os multiprocessadores. É importante compararmos que no pipeline escalar as instruções são realizadas ao mesmo tempo, entretanto em estágios diferentes. Por outro lado, em uma arquitetura superescalar há a duplicidade de hardware, na qual duas instruções podem ser executadas simultaneamente sem precisarem estar em estágios diferentes.

4.

- **Dependência de dados**: ocorre quando uma instrução pode ser obtida, decodificada, mas não executada, tendo que esperar que a instrução processadora seja executada, pois necessita de dados gerados pela mesma. Havendo essa dependência, a última instrução é atrasada em relação à primeira por tantos ciclos de clock necessários para remover a dependência.

- **Dependência de desvio**: é gerada com a ocorrência de desvios em uma sequência de instruções, fazendo com que esse fator complique a operação do pipeline. Essas instruções que ocorrem depois de um desvio possuem uma dependência procedural com o desvio e não podem ser executadas até que o mesmo seja executado. Uma instrução deve ser decodificada, pelo menos parcialmente, antes que a instrução seguinte possa ser buscada. Isso impede a busca simultânea de instruções, pedidas em uma pipeline superescalar.

- **Conflito de recurso (ULA)**: Um conflito de recurso ocorre quando duas ou mais instruções competem, ao mesmo tempo, por um mesmo recurso, como a ULA. Eles podem ser minimizados pela duplicação de recursos disponíveis para uso na máquina.

5.

Algumas dependências de saída podem surgir quando os valores nos registradores não refletem a sequência de valores estabelecida pelo fluxo do programa, resultando em atraso do estágio de pipeline. Uma solução para isso é a técnica de renomeação dos registradores por meio da alocação dinâmica de um novo registrador quando o valor

registrado for mudado. Ao renomear os registradores, adquire-se paralelismo, possibilitando que duas instruções que antes deveriam ser executadas em sequência possam agora ser executadas ao mesmo tempo utilizando registradores diferentes.

6.

Delayed branch e Otimização do branch são duas maneiras diferentes de diminuir os efeitos de branching em um pipeline. Sem eles, caso o branch seja realizado, o pipeline teria que descartar todo o processo de busca, decodificação, entre outros (que foram realizados antes da execução do branch).

A técnica de delayed branch atrasa a inicialização de instruções que devem ser feitas após a execução da instrução que possui o branch e coloca (número de estágios - 1) NOOPs no lugar, o que acaba diminuindo o desempenho. Essa abordagem faz com que funcione melhor num hardware mais simples, mas sobrecarrega o compilador;

A otimização do branch é uma técnica que busca prever um branch e “abaixa” (número de estágios - 1) instruções que não possuam algum tipo de dependência, nem interfiram no branch e que já seriam executadas normalmente antes da mudança de ordem, podendo então serem executadas sem problemas e sem “desperdiçar” ciclos.