Disciplina SSC0510
Arquitetura de Computadores

Grupo n° 9

Nome: Eduardo Eity Furuta	N° USP: 4940372
Nome: Mauricio Hitoshi Murakami	N° USP: 10295346
Nome: Guilherme da Silva Biondo	N° USP: 8124267
Nome: Amanda Lindoso Figueiredo	N° USP: 10784306

1ª Questão:

SISD: São máquinas de um fluxo único de instrução e fluxo único de dados (Single Instruction Single Data), essa classe de máquina são os computadores convencionais. As instruções são executadas em série.

SIMD: São máquinas com vários processadores e uma unidade de controle (Single Instruction Multiple Data), essa classe representa os processadores matriciais, paralelos e associativos. A unidade de controle envia a instrução para diversos processadores ao mesmo tempo e cada processador trabalha em um fluxo de dados diferente. Ex. Cambridge Parallel Processing Gamma II Plus.

MISD: São máquinas de múltiplas instruções e um único fluxo de dados. (Multiple Instruction Single Data). São máquinas que possuem vários processadores e cada processador recebe uma instrução diferente e o mesmo fluxo de dados é processado por todos os processadores. Ex. um pedal de efeito para guitarra elétrica, onde o sinal da guitarra recebe diversas transformações até a obtenção do efeito desejado na saída. Máquina para descriptografar dados.

MIMD: São máquinas com vários processadores e várias unidades de controle(Multiple Instruction Multiple Data). Cada processador pode receber instruções diferentes e dados diferentes. Eles podem trabalhar de forma síncrona ou assíncrona. Ex. computadores com vários núcleos e que trabalham com threads.

2ª Questão:
Memória compartilhada: é um tipo de arquitetura em que múltiplos processadores acessam a mesma memória. Nesse tipo de arquitetura a comunicação entre os processos é bem eficiente pois não existe a necessidade dos dados se movimentarem fisicamente. Além disso, a programação não difere muito da programação para um único processador.

Entretanto, esse tipo de memória deixa a cargo do desenvolvedor realizar sincronização para regiões compartilhadas. E dependendo da linguagem utilizada é um processo abstraído do desenvolvedor

Memória distribuída: é um tipo de arquitetura em que as tarefas são distribuídas para diversas máquinas e cada máquina possui memória exclusiva para o processador local. Nesse tipo de arquitetura existe uma vantagem que a memória é altamente escalável. Além disso, a forma de comunicação desse tipo de memória resolve o problema de sincronização que a memória compartilhada possui. O acesso aos dados de uma máquina por outra são feitas através de trocas de mensagens entre elas.

Porém, a programação feita na memória exige que o problema possua uma natureza paralela. E também é necessário uma distribuição de carga entre processadores, seja de maneira automática ou manual.

3ª Questão:

Processador com pipeline de operações: São processadores que conseguem aproveitar melhor os ciclos de operação do processador, cada instrução passa por diversos estágios antes de serem executadas pelo processador, em geral as instruções passam pelo estágio de Busca, Decodificação e Execução, os processadores com pipeline consegue executar mais de um estágio ao mesmo tempo aproveitando melhor o ciclo de tempo.

Processadores superescalares: São processadores capazes de utilizar múltiplos pipelines de instruções. As máquinas superescalares conseguem colocar mais de uma instrução em cada estágio de pipeline.

Processadores paralelos: são máquinas com múltiplos processadores (múltiplos núcleos de processador) trabalhando paralelamente, as tarefas podem ser executadas de forma independente em cada processador e cada processador possui seu pipeline de instruções.

4ª Questão:

No paralelismo existem algumas limitações, na dependência de dados por exemplo, uma instrução não pode ser executada se ela depender do resultado de outra instrução, nesse caso a instrução dependente precisa esperar que a instrução anterior seja executada primeiro. Para minimizar a dependência de dados, deve-se tentar colocar a execução da instrução dependente no final e deixar as instruções independentes no início para que estas aproveitem o paralelismo.
Na dependência do desvio, uma instrução depende de uma condição de outra instrução ser satisfeita, ou seja, pode ser que uma instrução não seja executada e portanto não necessita ser carregada no pipeline. Nesse caso pode-se atrasar as instruções dependentes do desvio com NOOP ou preencher o pipeline com instruções independentes antes do programa chegar na linha de instrução que depende do desvio condicional.
No conflito de recurso, duas instruções não podem acessar o mesmo recurso ao mesmo tempo, é possível minimizar esse conflito com pipeline de recursos ou a duplicação de recursos.

5ª Questão:

Dado que as instruções executadas serialmente são custosas em termos de tempo, a técnica de renomeação de registradores é utilizada para aumentar o paralelismo disponível, permitindo, dessa forma, que duas instruções (antes executadas serialmente) possam ser executadas de forma paralela (concorrente), pois utilizam registradores diferentes. A renomeação de registradores é baseada na técnica de tratamento de conflitos por recurso. A cada instante de tempo é associado um valor para um registrador, e esse valor é requerido pelas instruções. Quando um novo valor necessita ser armazenado em um registrador que já está em uso, um registro novo é criado. O acesso a esse valor (que deveria estar neste registrador), seguirá um processo de renomeação de registradores. As referências à registradores existentes nesta instrução serão revisadas passando a referir-se aos registradores que contém os valores requeridos.

6ª Questão:

A técnica do Delayed Branch visa atrasar o “preenchimento” do pipeline das instruções que dependem da condição da instrução anterior, as instruções que podem não ser executadas devido a condição da instrução anterior são atrasadas por instruções NOOP.

Ex. Uma máquina de 2 estágios (Busca e Execução)

Endereço	Instruções Normal	Delayed
100	LOAD X, A	LOAD X, A
101	ADD 1, A	ADD 1, A
102	JUMP 105	JUMP 106
103	ADD A, B	NOOP
104	SUB C, B	ADD A, B
105	STORE A, Z	SUB C, B
106		STORE A, Z

No exemplo acima, as instruções do endereço 103 em diante foram atrasadas. Assim, quando a instrução no endereço 102 estiver sendo executada o processador estará buscando a instrução no endereço 103, caso o JUMP da instrução no endereço 102 ocorra, o pipeline não precisa ser esvaziado pois foi preenchido com a instrução NOOP.

Na técnica de otimização do branch a instrução onde pode ocorrer o branch é trocada com instrução anterior. Essa troca só pode ocorrer se as instruções forem independentes.

Ex. Uma máquina com pipeline de 2 estágios (Busca e Execução)

Endereço	Instruções Normal	Otimizado

100	LOAD X, A	LOAD X, A
101	ADD 1, A	JUMP 105
102	JUMP 105	ADD 1, A
103	ADD A, B	ADD A, B
104	SUB C, B	SUB C, B
105	STORE A, Z	STORE A, Z

As instruções do endereço 101 e 102 foram trocadas, desta forma quando a instrução JUMP estiver sendo executada a instrução ADD 1, A está sendo buscada, e independemente do JUMP ocorrer ou não, a instrução no endereço 102 pode ser executada, evitando assim o esvaziamento do pipeline.