Disciplina SCE541
Arquitetura de Computadores

Grupo nr.: 8

Nome: Thiago Ribeiro Goveia Nº USP: 10835942

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

- SISD - Single Instruction Single Data

Nesta arquitetura existe apenas um fluxo de instruções e um fluxo de dados, por fluxo de instruções, entendemos o processo de leitura das instruções da memória pela máquina de controle que as decodifica e envia comandos para a unidade de processamento. Fluxo de dados é aquele que acontece entre a unidade de processamento e a memória, em que essa carrega e salva dados da mesma. A arquitetura SISD é encontrada em computadores convencionais e uma forma de paralelismo só pode ser atingida através do uso de pipeline. Um exemplo de computador SISD é o arduino.

- SIMD - Single Instruction Multiple Data

Nesta arquitetura existe apenas um fluxo de instruções que é aplicado para processar múltiplos fluxos de dados. Máquinas desse tipo possuem uma unidade de controle que se comunica com diversas unidades de processamento que trabalham em paralelo, os comandos passados à cada uma delas são iguais, porém os dados processados podem ser distintos. Essas arquiteturas assim são úteis para realizar operações matriciais, nas quais a mesma operação é realizada em várias células contendo dados diferentes.

- MISD - Multiple Instruction Single Data

Nesta arquitetura existem múltiplos fluxos de instruções e apenas um fluxo de dados. Essas máquinas possuem várias unidades de processamento e unidades de controle, que operam sobre o mesmo conjunto de dados. Isso é atingido ao se carregar instruções diferentes em cada unidade de comando e aplicá-las a um conjunto de dados em sequência, retornando o resultado à memória no final. Exemplo de máquina que utiliza essa arquitetura é um filtro passa baixa, que aplica filtros sequenciais à uma corrente para que apenas frequências abaixo de um threshold possam passar.

- MIMD - Multiple Instruction Multiple Data

Nesta arquitetura existem múltiplos fluxos de instruções e múltiplos fluxos de dados. Essas máquinas podem realizar operações completamente independentes, trabalhando síncrona ou assincronamente, permitindo o conceito de threads, onde cada conjunto de unidade de controle e unidade de processamento opera sob um fluxo de instruções e de dados distinto, o que as diferencia das máquinas de arquitetura MISD, na qual apenas dois dos n conjuntos estão ligados diretamente aos barramentos de dados da memória e são capazes de iniciar/finalizar o fluxo de dados. Processadores modernos são construídos na arquitetura MIMD.

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

- Memória compartilhada: é um modelo de acesso à memória no qual um a memória é compartilhada entre múltiplos processadores de forma concorrente. Uma vantagem imediata deste modelo é que a leitura e escrita de\em um local da memória é simplificado, uma vez que todos os processadores têm acesso ao mesmo. As tarefas executadas pelos processadores realizam operações de escrita e leitura na mesma memória, realizando-se um controle de acesso para manter a coerência dos dados, esse trabalho de sincronização é realizado pelas unidades de controle ligadas aos processadores. A sincronização é realizada através de trocas de mensagens entre os processadores através de barramentos, para que a sinalização da utilização de recursos de memória seja realizada, ficando o usuário responsável por ela, através de linguagens de programação paralelas. Isso permite grande agilidade na comunicação entre os processadores, com limitação de escalabilidade pelo número de caminhos de memória. Exemplos de arquiteturas com memória compartilhada é o processador i7 da Intel, no qual os núcleos, através de threads, acessam uma memória compartilhada concorrentemente.
- Memória distribuída: nesta arquitetura a memória está fisicamente distribuída entre as unidades e o seu acesso a está restrito ao processador a ela conectado. A comunicação entre entre os processadores é feita através de mensagem numa rede de comunicação. A leitura e escrita de dados em meḿorias externas a um processador é realizada através de solicitações que são executadas pelo processador conectado à memória que é dona do dado em questão. Isso gera uma latência maior de comunicação quando comparados a memória compartilhada, porém tem menor restrições de escalabilidades que estão mais relacionadas ao custo e infraestrutura do que ao espaço físico. Exemplos dessa arquitetura são os MPPs como o Intel Paragon em que os processadores estão numa mesma placa mãe. Clusters de computadores também seguem essa arquitetura, porém sua escalabilidade é limitada por gasto energético e são de difícil programação para gestão dos dados e instruções.

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

1) Processador com pipeline de operações
2) Processadores Superescalares
3) Processadores Paralelos

3.1) Processador com pipeline de operações

Um processador que possui um pipeline de operações utiliza-se do fato de que determinadas operações realizadas na execução de uma instrução utilizam recursos diferentes do processador, o que permite que elas sejam executadas em paralelo, dentro de um ciclo da CPU, dando a impressão de que a cpu pode executar as instruções de forma paralela quando, na verdade apenas um fluxo de instruções existe no processador. Isso é atingido dividindo as operações da instrução, por exemplo num pipeline de três estágios: busca, decodificação e execução em ciclos da CPU. Assim, a cpu pode buscar uma instrução enquanto decodifica e executa outras duas simultaneamente. A utilização de pipelines permite grandes ganhos de velocidade de processamento, chegando a 20 vezes da velocidade normal do processador. Esse ganho, porém, pode ser prejudicado por jumps e branches nas instruções, que causam o preenchimento do pipeline com instruções que podem não ser executadas, o código carregado que não será executado demanda o esvaziamento e um novo preenchimento do pipeline, desperdiçando ciclos do processador.

3.2) Processadores Superescalares

Processadores superescalares possuem várias pipelines que processam instruções independentes. Cada um desses pipelines podem processar várias instruções ao mesmo tempo, permitindo múltiplos fluxos de instruções. A organização das instruções entre as pipelines é feita quanto ao tipo de recurso demandado por essas instruções, que são divididas e enfileiradas considerando esses recursos, para então serem enviadas aos pipelines. Um grande ganho de desempenho é obtido ao executar instruções independentes em ordens que sejam ótimas ao uso dos recursos do processador, permitindo que instruções de execução mais rápida não sejam proteladas por instruções mais lentas. Uma limitação a este recurso é a dependência entre instruções, assim como a existência de branches como no pipeline regular.

3.3) Processadores Paralelos

Máquinas com processadores paralelos podem atingir o paralelismo verdadeiro através da utilização de threads. Como cada processador possui seus recursos isolados, apenas deve ser realizado controle de acesso a memória, ficando a cargo de cada processador lidar com as instruções. Isso permite que vários fluxos de instruções sejam estabelecidos paralelamente. Adicionalmente, processadores paralelos podem possuir uma ou múltiplas pipelines de operações para lidar com o fluxo de instruções a ele direcionado utilizando-se dos benefícios do pipeline.

4) 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.

A execução de instruções em paralelo enfrenta dificuldades relacionadas à disponibilidade de dados e recursos para uso pelas múltiplas instruções sendo processadas. A dependência de dados ocorre quando uma instrução está modificando um dado que será utilizado por outra instrução em sequência, isso causa uma dependência entre elas, ou seja, a segunda instrução só pode ser executada após a execução da primeira. A dependência de dados é dita verdadeira quando a próxima instrução tem como entrada a saída da instrução anterior.

Dependências de desvio ocorrem quando existe um desvio condicional que faz com que instruções já carregadas no pipeline não sejam executadas, demandando seu esvaziamento e novo preenchimento, perdendo assim ciclos do processador que não são utilizados para execução de instruções. Consequentemente, instruções com dependência de desvio não podem ser executadas até a execução do desvio.

Conflito de recursos ocorre quando instruções demandam o mesmo recurso para sua execução, como por exemplo uma ULA, ou realizar um acesso à memória ou utilizar um barramento E/S. Pode ser minimizado com a aplicação de pipeline de recursos, permitindo que as operações de cada recurso sejam executadas com paralelismo em múltiplos pipelines, ou ainda tento múltiplos dos recursos do processador disponíveis para uso paralelo.

Atrasos causados por dependências podem ser mitigados trocando a ordem de execução das instruções independentes, de forma a minimizar o tempo morto dos recursos do processador. Pode-se também utilizar mais registradores para minimizar a chance de que um registrador seja usado por mais de uma instrução para fins distintos em um curto espaço de tempo. Finalmente, técnicas de previsão de desvio como delayed branch e otimização podem ser utilizadas.

5) Explique renomeação de registradores

Renomeação de registradores é uma técnica utilizada para lidar com problema de antidependência em pipelines, no qual uma instrução que foi iniciada antes do que seria normalmente destrói um valor que seria utilizado por uma instrução prévia que porventura veio a ser executada depois. Neste caso, pode-se renomear o registrador modificado pela segunda instrução para um novo registrador, preservando o valor do registrador original, que, então, poderá ser utilizado pela instrução prévia.

Sem renomeação

Processador 1	Processador 2
I1: r3 <- r2 + r5	I2: r4 <- r3 + 1
I3: r3 <- r4+1	I4: r7 <- r3 + r5

Com renomeação

I1: r13 <- r2 + r5	I2: r4 <- r13 + 1
I3: r3 <- r4+1	I4: r7 <- r3 + r5

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

A técnica Delayed Branch é uma técnica de otimização de pipeline utilizada para lidar com desvios. Ela consiste em inserir instruções NOOP após os branches para que essas possam ser executadas pelo pipeline que executou um pré-fech, quer o desvio seja tomado, quer não. Esta técnica não confere de fato uma melhoria de velocidade, pois a execução de NOOPs toma o tempo que seria utilizado na reorganização do pipeline, portanto ela é tão ruim quanto caso o desvio seja tomado e é pior caso ele não seja tomado. O seu benefício é observado quanto considera-se que esta técnica permite que a máquina de controle ignore a existência de desvios, dispensando a aplicação de algoritmos de predição de desvio, simplificando seu projeto.

A técnica de otimização de branch consiste em reorganizar as instruções de forma que a logo após o desvio, sejam carregadas instruções independentes que, normalmente, viriam antes dele, para que estas sejam executadas após a execução do desvio, impedindo que seja feito o prefetch de instruções condicionadas aos desvio.

Ambas as técnicas podem ser implementadas por compiladores para otimizar o código para a execução em pipelines.