Lista 2 de Infraestrutura de Hardware

1. a) Podem ocorrer três tipos de conflito. O primeiro é o conflito estrutural, quando duas instruções querem acessar o mesmo componente de hardware, como por exemplo, uma instrução quer salvar um valor na memória, e outra quer ler uma posição da memória. O segundo é o conflito de dados, que é quando uma instrução precisa de um dado que ainda está sendo escrito em outra instrução, como por exemplo t1 = t2 + t3 e logo depois t4 = t1 + t3. O terceiro é o conflito de controle, que é quando a execução ou não de uma instrução depende de uma decisão que está sendo tomada, por exemplo, a linha que vem logo abaixo de um beq.

b) Para resolver o conflito estrutural, pode-se replicar recursos, como por exemplo, uma ULA para dados e outra para instruções. Para conflito de dados, se congela a pipeline até que o dado esteja disponível, e pode se aproveitar também de *fowarding,* que é usar o resultado da ula antes mesmo de escrever em um registrador. Para conflito de controle, pode-se congelar a pipeline até a decisão ser feita (mas isso causa perda de desempenho), uma solução melhor é ir executando as instruções subsequentes até saber se elas devem ser realmente feita ou não, caso negativo, zera-se os sinais de controle e cancela a instrução; também é possível usar delayed branch, que é quando se coloca logo após o branch instruções que vão ser executadas independente da decisão, o que garante menos desperdício de desempenho, já que as instruções (do delay slot) vão ser executadas até o final e não haverá *flush*.

1. a) Tanto no monociclo quanto no multiciclo, somente uma instrução está no datapath em um instante. Isso significa que enquanto a instrução está em um determinado estágio, todos os outros recursos ficam ociosos. A pipeline tem como objetivos ampliar a utilização desses recursos, assim há menos ócio e o throughput aumenta. O lado ruim do pipeline é que deixa o hardware mais complexo e também aumenta seu custo, tanto pela complexidade maior, como também pela replicação de recursos.

b) A arquitetura do MIPS dá suporte ao desvio retardado (delayed branch), que, como explicado na primeira questão, re-arruma o código e coloca uma instrução independente do branch para ser executada enquanto o branch está sendo avaliado.

c) O que pode alterar o desempenho do pipeline são os eventuais conflitos, seja de dado ou de controle, que pode fazer a pipeline congelar por alguns ciclos ou executar parte de instruções desnecessariamente (flush).

1. Multi-thread tem como objetivo a melhor utilização de recursos executando várias threads virtualmente de forma simultânea no mesmo usar menos área do chip. Os tipos possíveis de multi-thread são fine-grain: onde ocorre troca de thread a cada instrução num escalonamento round-robin, é bom porque esconde a perda de throughput em paradas longas e curtas, mas é ruim porque pode fazer muitas trocas para uma thread que está ociosa; course-grain: troca de threads somente em paradas longas, evita troca desnecessárias, mas tem que anular o pipeline no caso de uma dessas paradas; e smt: é multi-thread para processadores superescalares, tem paralelismo a nível de thread e a nível de instrução, porém é mais custoso.

1. a) Superpipeline é uma melhora do pipeline, na qual os estágios são divididos em subestágios. Isso proporciona um período menor, logo uma frequência e um throughput maior. Porém, causa maior dependência de dados e maior custo de hardware.

b) Superescalar é um processador onde um grupo de instruções podem ser executadas no mesmo estágio ao mesmo tempo. Precisa de um custo de hardware maior, porém proporciona paralelismo e por consequência, um maior throughput. No caso de ser um superescalar de fato (e não um VLIW), a execução ainda é atrasada porque o processador ainda tem que escolher quais instruções podem ser executadas ao mesmo tempo.

1. Processadores multicore são processadores com vários núcleos atuando ao mesmo tempo. A comunicação entre os processadores pode ser feita de duas formas diferentes: com memória compartilhada ou passagem de mensagem. No primeiro caso, há variáveis compartilhadas entre os processadores, e só um processador por vez pode acessar a uma dessas variáveis. Para isso é utilizado o semáforo (lock), com o comando ll, o processador vê se o semáforo está livre, caso negativo segue tentando acessar, caso positivo acessa a região, usa o comando sc =1 para dizer q a região está ocupada, faz as alterações que deseja e depois sc=0 para liberar o semáforo. Com passagem de mensagem os processadores compartilham explicitamente os dados e eles se comunicam através de primitivas *send* e *receive.*