Lista 2 - AOC 2019.2
Aluno: Markus Kaul M. Gerrits

1. Quais as diferenças de um processador multiciclo em relação a um uniciclo?

   Processadores uniciclo são baseados em um único ciclo de clock, grande o suficiente para acomodar todas as instruções a serem consideradas. Toda a instrução começa sua execução em uma transição ativa do clock e completa a execução na próxima transição ativa do sinal do clock. Assim sendo, todas as instruções gastam o mesmo tempo para serem executadas, tempo este que vai corresponder ao tempo gasto na execução da instrução mais demorada, e que deve obrigatoriamente ser igual ao ciclo do clock.

   Apesar de muito simples de entender, este esquema não se revela prático, pois é muito mais lento do que uma outra implementação que permite que classes diferentes de instruções gastem exatamente o tempo necessário às suas execuções. A vantagem desse tipo de implementação está no fato de que os tempos de execução de cada classe de instrução variam substancialmente.

   Nos ***processadores multiciclos*** cada passo de execução gasta um período do clock. A implementação multiciclo permite que uma unidade funcional seja utilizada mais de uma vez por instrução, uma vez que ela está sendo usada em ciclos diferentes do clock. Esta possibilidade de compartilhamento pode ajudar a reduzir a quantidade de hardware necessário à implementação. Em resumo, as principais vantagens da implementação multiciclo são a possibilidade de fazer com que instruções sejam executadas em quantidades diferentes de períodos do clock, e a capacidade de compartilhar unidades funcionais dentro do espaço de tempo necessário à execução de uma única instrução.

   Ao final de um ciclo de clock, todos os dados que precisarem ser usados em ciclos subsequentes devem ser armazenados em um elemento de estado. Os dados a serem usados em outras instruções devem ser armazenados em elementos de estado visíveis ao programador, ou seja, no banco de registradores, no PC ou na memória.

2. Quais as modificações necessárias em um processador multiciclo simples para que se introduza a função de pipeline?

   Pipeline é uma técnica de implementação em que várias instruções são sobrepostas na execução. Hoje, a técnica de pipeline é fundamental para tornar os processadores mais rápidos.

Para que um processador multiciclo possa ter a função de pipeline é preciso fazer algumas mudanças, e a primeira mudança necessária é a separação da memória em duas partes independentes. Uma parte será utilizada apenas para instruções. e outra apenas para os dados. Isso é necessário para que a etapa IF acesse a memória para buscar a próxima instrução, ao mesmo tempo em que a MEM acessa a memória para salvar o resultado de outra instrução anterior. Se houvesse apenas uma memória para dados e instruções, isso não seria possível.

Outra mudança importante foi a adição de memórias intermediárias entre cada etapa. Essas memórias são utilizadas para armazenar o resultado da etapa anterior e passá-lo para a etapa posterior no ciclo seguinte. Elas são necessárias porque as etapas não executam necessariamente sempre na mesma velocidade. Se uma etapa for concluída antes da etapa seguinte, seu resultado deve ser guardado nessas memórias para aguardar que a etapa seguinte conclua o que estava fazendo. Só então ela poderá receber o resultado da etapa anterior.

3. No programa abaixo, relacione as dependências (dados, WAR, WAW e outros) e conflitos existentes para execução em um processador MIPS usando pipeline.

Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14
div.d F1, F2, F3	IF	ID	EX	WB										
	bolha	bolha	bolha	bolha	bolha	bolha	bolha							
sub.d F4, F5, F1				IF	ID	EX	WB							
							bolha	bolha	bolha	bolha				
s.d F4, 4(F10)							IF	ID	EX	MEM				
add.d F5, F6, F7								IF	ID	EX	WB			
								bolha	bolha	bolha	bolha			
div.d F4, F5, F6											IF	ID	EX	WB

1º Conflito => RAW: primeiro div.d e sub.d com o registrador F1;
2º Conflito => RAW: sub.d e s.d com o registrador F4;
3º Conflito => RAW: add.d e segundo div.d com o registrador F5;
4º Conflito => WAW: sub.d e segundo div.d com o registrador F4;
5º Conflito => WAR: sub.d e add.d com o registrador F5;

4. Descreva os seguintes conceitos:

a) *Write though:* É relativo à quando a escrita de dados é feita simultaneamente na memória Cache, quanto no endereço correspondente na memória principal, fazendo que dessa forma

não se atrapalhe com dados desatualizados, pois sempre que houver uma alteração do valor a memória principal será atualizada, contudo essa técnica traz consigo um maior tempo, pois toda vez ter que atualizar o valor na memória principal tem um gasto.

b) *Write back:* É a técnica que o processador faz a escrita diretamente no Cache e fica como responsabilidade do sistema fazer a atualização dos dados na memória principal, dessa forma se usa menos tempo, pois não tem que ficar atualizando a memória principal a cada alteração, mas pode acontecer que o processador acabe operando com dados desatualizados, por que o sistema não mandou a instrução para a atualização dos dados na memória principal.

c) *Localidade Temporal:* Se trata do reconhecimento que um dado que foi acessado recentemente têm mais chance de ser usado novamente, do que um dado que já foi usado há muito tempo, dessa forma os dados recentes que estão na RAM são copiados para o Cache.

d) *Localidade Espacial:* Se trata do reconhecimento há uma maior chance de se acessar um dado de posições próximas a um dado já acessado, por exemplo programas sequenciais tendem a acessar endereços de memória próximos e com isso quando eu acesso uma instrução a instrução com mais chance de ser acessada será a próxima, com isso as instruções próximas as que estão sendo usadas ficam armazenadas na memória cache.