	<p style="text-align: center;"> <Laboratório VHDL> por <Bruno Rodrigues > - 2018.2 Boa vista, 06/12/2018 </p>
--	--

Lista 2

Bruno Rodrigues Caputo (brunorcx@hotmail.com)

DCC301-Arquitetura e Organização de Computadores 2018.2-Turma 01

Universidade Federal de Roraima

DCC -Departamento de Ciência da Computação - Bloco V

Campus Universitário do Paricarana - Aeroporto

69310-000 Boa vista, RR


1) Quais as vantagens de um processador multiciclo em relação a um uniciclo?

R-Considerando os cinco passos de execução de uma instrução por ciclo de clock temos: IF -Busca, ID a Decodificação, EX como execução, MEM a Memória e WB como Escrita de dados. Para que um processador consiga executar uma instrução do tipo R no uniciclo ele deve se basear na instrução que irá utilizar o maior número de unidades em série, assim, mesmo que consiga efetuar a execução da instrução em menor número de clock terá de aguardar com tempo ocioso. O multiciclo consegue separar as instruções de acordo com as etapas, então uma instrução do tipo J não iria precisar esperar a execução de todos os componentes que um *store* necessitaria.

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

R-Deve-se considerar a inicialização das outras instruções enquanto se executa a primeira, considerando claro, componentes que não estejam sendo utilizados e que não tenham dependências anteriores, não se pode executar uma instrução do tipo add R1,R2,R3 Sem o ter o valor do registrador R2 ou R3, ou seja, caso R2 ou R3 esteja sendo utilizado em uma instrução anterior o processador é preciso esperar e respeitar o resultado da instrução, além de conflitos estruturais ou de controle.

3) Considerando o pipeline do MIPS (simples com MEM compartilhada para instrução e dados) e uma iteração de loop conforme o trecho de programa abaixo, relacione os conflitos que podem ocorrer e seus consequentes stalls. Qual o speedup (por iteração) para o programa em relação à versão sem pipeline?

	<p style="text-align: center;"><Laboratório VHDL> por <Bruno Rodrigues > - 2018.2 Boa vista, 06/12/2018</p>
--	---

Loop: subi \$t2, \$t2, 4	1
lw \$t1, 0(\$t2)	2
add \$t3, \$t1, \$t4	3
add \$t4, \$t3, \$t3	4
sw \$t4, 0(\$t2)	5
beq \$t2, \$0, loop	6

R- Logo na linha 2 temos a utilização do registrador \$t2 em *lw*, porém na linha acima foi executada a instrução *sub*, como é preciso é armazenado em \$t2 a subtração teremos um stall para executar o *load*.

Novamente na linha 3 temos a utilização do registrador \$t1 na soma, porém na linha acima \$t1 é preciso aguardar o carregamento da palavra

Na linha 3 usamos o registrador \$t3 para armazenar a soma e na linha 4 teremos de esperar a o resultado da soma para fazer uma nova adição.

Na linha 4 usamos o *add* para armazenar a soma e logo embaixo utilizamos o *sw* para armazenar no endereço 0(\$t2).

A versão sem pipeline irá executar as instruções em ordem IF-ID-EX-MEM-WB, enquanto que com pipeline teremos um ganho em já começar próxima instrução IF já na execução da ID inicial.


4) No programa abaixo, relacione as dependências (dados, WAR, WAW e outros) existentes.

div.d	F1,	F2,	F3	1
sub.d	F4,	F5,	F1	2
s.d	F4,	4(F10)		3
add.d	F5,	F6,	F7	4
div.d	F4,	F5,	F6	5

R- Temos um WAW na linha 2 com F1 que foi utilizado na linha anterior. Na linha 3 temos um erro de dados, foi utilizado F4 duas vezes o mesmo registrador. Na linha 5 temos um WAW com F5 sendo utilizando na linha anterior.

5) Em relação a memória cache. Um computador tem CPI 1 quando todos os acessos à memória acertam no cache. Loads e Stores totalizam 50% das instruções. Se a penalidade por miss é de 25 ciclos e o miss rate é 2%, qual o desempenho relativo se o computador acertar todos os acessos?

R- tempo de execução de CPU todos os acertos

	<p style="text-align: center;"> <Laboratório VHDL> por <Bruno Rodrigues > - 2018.2 Boa vista, 06/12/2018 </p>
--	--

$$\begin{aligned}
 &= (\text{CPU ciclos por clock} + \text{ciclos de stalls por memória} \times \text{ciclo clock}) \\
 &= (\text{IC} \times \text{CPI} + 0) \times \text{Ciclo de clock} \\
 &= \text{IC} \times 1 \times \text{Clock cycle}
 \end{aligned}$$

Na situação real

Ciclos de stalls por memória

$$\begin{aligned}
 &= \text{IC} \times \text{acesso de memória/instrução} \times \text{taxa e falha} \times \text{penalidade de falha} \\
 &= \text{IC} (1 + 0,5) \times 0,02 \times 25 \\
 &= \text{IC} \times 0,75
 \end{aligned}$$

Resultado:

$$\begin{aligned}
 &\text{Cache de tempo de execução de CPU/ execução de CPU} \\
 &= 1,75 \times \text{IC} \times \text{Ciclo clock} / 1,0 \times \text{IC} \times \text{Clock cycle} \\
 &= 1,75
 \end{aligned}$$

6) Descreva os seguintes conceitos:

a) Write through

b) Write back


c) Localidade Temporal

d) Localidade Espacial

a) É um método de armazenamento onde o dado é escrito no cache e na memória correspondente ao mesmo tempo, permitindo uma retirada rápida quando necessário enquanto o mesmo dado na memória principal garante que nada será perdido com uma falha de energia.

b) acelera a performance uma vez que atualizações são normalmente escritas exclusivamente no cache e carregadas na memória principal em intervalos específicos ou dentro de certas condições.

c) e d) No princípio da localidade os mesmos valores ou lugares de armazenamentos relacionados são frequentemente acessados dependendo do tipo de padrão de acesso à memória. Temos dois tipos básicos Localidade Temporal e Localidade Espacial. Temporal refere ao reuso de dados específicos ou recursos dentro um tempo de duração pequeno. Espacial se refere ao uso de elementos de dados dentro lugares de armazenamentos relativamente próximos.

 <p>UFRR</p>	<p><Laboratório VHDL> por <Bruno Rodrigues > - 2018.2 Boa vista, 06/12/2018</p>
--	---

Bibliografia

Retirado do sítio "<https://whatis.techtarget.com/definition/write-through>"



<Laboratório VHDL>
por <Bruno Rodrigues > - 2018.2
Boa vista, 06/12/2018