O início do loop se dá com dá com o carregamento do próximo valor do loop (lw x12 0(x9)).  
Em seguida ele realiza um Add com dependendo do valor carregado na instrução anterior o que gera um Stall. E no final para que o branch possa tomar a decisão de continuar no loop ou continuar com as instruções normais, é realizado dois stalls, pois o Branch necessita do valor calculado na instrução anterior. A imagem abaixo demonstra o esquema gerado pelo simulador Ripes.

1. Fetch instruction from memory.

2. Read registers and decode the instruction.

3. Execute the operation or calculate an address.

4. Access an operand in data memory (if necessary).

5. Write the result into a register (if necessary).

Hazards:

KATZ (1996) descreve Hazards como limitações do pipeline que previne a próxima instrução de ser executada no tempo de clock designada a ela. Existem três tipos de Hazards:

Hazard Estrutural:

Definição: Quando uma instrução, devido ao conjunto de intruções que precedem ou sucedem essa instrução, a impedem de utilizar a estrutura de dados, ou seja de executar o procedimento que devia no ciclo de clock adequado. Ou seja duas instruções tentando utilizar a mesma estrutura ao mesmo tempo.

Exemplo: Um exemplo para Hazard Estrutural é o levantado por Patterson e Hennessy (2018), onde é criado uma situação hipotética onde só haja uma estrutura de memória no processador, e tenhamos uma instrução que acessa a memória no 4º ciclo e esse ciclo é executado simultâneo a uma outra instrução que está executando seu primeiro ciclo, buscando na memória também, a tentativa de uso simultaneo da mesma estrutura no mesmo ciclo de clock gera entao o Hazard Estrutural. E nesse caso os próprios Patterson e Hennessy (2018) explicam que o conjunto de instruções do pipeline foi projetado para evitar esse tipo de instruções, tornando simples para os designers evitarem esse tipo de hazard, como no caso a cima que a utilização de duas memórias evita esse erro.

Hazard de dados:

Definição: Como esclarecido por KATZ (1996) hazard de dados são causados por instruções, que para serem executadas, dependem de valores gerados por instruções que ainda estão sendo processadas.

Exemplo: Um exemplo simples é a execução de duas instruções do Tipo-R consecutivas, onde o resultado gerado na primeira é utilizado no processamento para o resultado da segunda. Uma opção é deixar o compilador ordenar a execução das instruções de forma a evitar a execução sequencial de intruções interdependentes, porém essas dependências são muito frequentes e uma solução simples para situações como o caso apresentado a cima no exemplo, é a criação de encaminhamentos. Encaminhamentos é a adição de hardware (barramentos) de forma que permite assim que calculado o valor da instrução anterior, ele já esteja disponível para o uso na próxima instrução antes mesmo que ele esteja disponível no banco de registradores ou na memória.

Hazard de Controles:

Definição: Centoducatte(1998) explica que hazard de controle, são problemas ocasionados pela execução de instruções de desvio. Ou seja, quando uma instrução necessita tomar uma execução enquanto outras instruções continuam executando.

Exemplo: O exemplo padrão é uma instrução branch, que o tempo entre o início da instrução e executar a comparação e decida realizar o jump, toma 4 ciclos de clock, que ao realizar isto a instução seguinte terá terminado terceiro ciclo, logo se o jump não for para a próxima instrução uma inconsitência terá sido gerada. Para solucionar esse tipo de problema é sempre necessário criar stalls após o branch. Em casos normais 2 Stalls serão necessários para garantir a estabilidade do sistema, no entanto esse valor pode ser otimizado com a implementação de um hardware auxiliar para tentar prever a ação do branch, antes mesmo de ser executado, o que em casos positivos economizaria 1 Stall ao processo.

Paulo C. Centoducatte 1998 Pipeline

Randy H. Katz 1996 Lecture 7: Introduction to Pipelining, Structural Hazards, and Forwarding

David A. Patterson John L. Hennessy 2018 Computer Organization and Design: THE HARDWARE/SOFTWARE INTERFACE