**PIPELINING AND PIPELINE HAZARDS**

1. No forwarding with optimization of branches

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| LD R1, 0(R2) | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DADDI R1, R1, 1 |  | F | D | s | s | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
| SD 0(R2), R1 |  |  | F | s | s | D | s | s | X | M | W |  |  |  |  |  |  |  |  |  |
| DADDI R2, R2, 4 |  |  |  |  |  | F | s | s | D | X | M | W |  |  |  |  |  |  |  |  |
| DSUB R4, R3, R2 |  |  |  |  |  |  |  |  | F | D | s | s | X | M | W |  |  |  |  |  |
| BNEZ R4, loop |  |  |  |  |  |  |  |  |  | F | s | s | D | s | s | X | M | W |  |  |
| LD (2) |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | F | D | X | M | W |

Steps taken to achieve this is:

* The first DADDI must wait until the previous LD gets to the Write Back stage to obtain the value of R1.
* SD must wait until the first DADDI computes the value of R1 and reaches the Write Back stage (no forwarding).
* The second DADDI depends on the value of R2 which was not changed earlier so it proceeds normally.
* DSUB must wait until the second DADDI computes the value of R2 and reaches the Write Back stage.
* BNEZ must wait until DSUB computes the value of R4 and reaches the Write Back stage.
* Then LD from the next iteration is re-fetched after the branch is resolved as taken in the branch ID stage
* so the iterations that the loop executes is 0 to 98
* Iteration i begins in , the last iteration takes 18 cycles to complete so the total number of cycles for the whole loop will be

1. Forwarding and branch predict as not taken

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| LD R1, 0(R2) | F | D | X | M | W |  |  |  |  |  |  |  |  |  |
| DADDI R1, R1, 1 |  | F | D | s | X | M | W |  |  |  |  |  |  |  |
| SD 0(R2), R1 |  |  | F | s | D | X | M | W |  |  |  |  |  |  |
| DADDI R2, R2, 4 |  |  |  |  | F | D | X | M | W |  |  |  |  |  |
| DSUB R4, R3, R2 |  |  |  |  |  | F | D | X | M | W |  |  |  |  |
| BNEZ R4, loop |  |  |  |  |  |  | F | D | s | X | M | W |  |  |
| LD (2) |  |  |  |  |  |  |  | F | s | F | D | X | M | W |

Steps taken to achieve this is:

* The first DADDI still must wait until LD gets to the Write Back stage to obtain the value of R1 but now stall is implemented in the Execution stage and the value is forwarded directly to Execution stage.
* SD now waits until the first DADDI computes the value of R1 and forwards the value from Execution to Memory.
* The second DADDI depends on the value of R2 which was not changed earlier so it proceeds normally.
* DSUB now gets the value of R2 forwarded by the second DADDI and does not need to wait.
* BNEZ now gets the value of R4 forwarded by DSUB but needs to wait till DSUB reaches Execution stage.
* LD from the next iteration is re-fetched after the branch is resolved as mispredicted in the branch ID stage.
* Iteration i begins in , the last iteration takes 12 cycles to complete so the total number of cycles for the whole loop will be

1. Normal forwarding with simple instruction scheduling and branch delay slot

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| LD R1, 0(R2) | F | D | X | M | W |  |  |  |  |  |  |
| DADDI R2, R2, 4 |  | F | D | X | M | W |  |  |  |  |  |
| DSUB R4, R3, R2 |  |  | F | D | X | M | W |  |  |  |  |
| DADDI R1, R1, 1 |  |  |  | F | D | X | M | W |  |  |  |
| BNEZ R4, loop |  |  |  |  | F | D | X | M | W |  |  |
| SD -4(R2), R1 |  |  |  |  |  | F | D | X | M | W |  |
| LD (2) |  |  |  |  |  |  | F | D | X | M | W |

Steps taken to achieve this is:

* Move the second DADDI to the load delay slot, move the DSUB up to after the second DADDI and move SD to the branch delay slot adjusting the offset to SD -4(R2), R1.
* The first DADDI does not wait as it is now separated by one extra cycle from LD.
* The BNEZ now does not wait as it is separated by extra cycle from DSUB.
* SD now fills the branch delay slot.
* LD from the next iteration is properly fetched as the branch is resolved in time.
* Iteration i begins in , the last iteration takes 10 cycles to complete so the total number of cycles for the whole loop will be