**Computer Architecture Lab 2**

**Name: BRYSON KATUU**

**Reg No: SCT212-0205/2021**

**Problem**

Use the following code fragment:

loop: LD R1,0(R2)

DADDI R1, R1,1

SD 0(R2), R1

DADDI R2,R2,4

DSUB R4, R3,R2

BNEZ R4, loop

Assume that the initial value of R3 is R2+396. Throughout this exercise use the classic RISC five-stage integer pipeline in H&P. Specifically, assume that: 1) branches are resolved in the second stage of the pipeline, 2) there are separate instruction and data memories; 3) all memory accesses take 1 clock cycle.

a. Show the timing of this instruction sequence for the RISC pipeline without any forwarding or bypassing hardware but assuming a register read and a write in the same clock cycle “forwards” through the register file. Assume that the branch is handled by flushing the pipeline. If all memory references take 1 cycle, how many cycles does this loop take to execute?

b. Show the timing of this instruction sequence for the RISC pipeline with normal forwarding and bypassing hardware. Assume that the branch is handled by predicting it as not taken. If all memory references take 1 cycle, how many cycles does this loop take to execute?

c. Assume the RISC pipeline with a single-cycle delayed branch and normal forwarding and bypassing hardware. Schedule the instructions in the loop including the branch delay slot.

You may reorder instructions and modify the individual instructions operands, but do not undertake other loop transformations that change the number or opcode of the instructions in the loops. Show a pipeline timing diagram and compute the number of cycles needed to execute the entire

**SOLUTION.**

* **F = Fetch**
* **D = Decode**
* **X = Execute**
* **M = Memory**
* **W = Writeback**
* **s = Stall (wait because of a hazard)**

1. **No forward and branching optimization**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| LD | F | D | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DADDI |  | F | D | S | S | X | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
| SD |  |  | F | S | S | D | S | S | X | M | W |  |  |  |  |  |  |  |  |  |
| DADDI |  |  |  |  |  | F | S | S | D | X | M | W |  |  |  |  |  |  |  |  |
| DSUB |  |  |  |  |  |  |  |  | F | D | S | S | X | M | W |  |  |  |  |  |
| BNEZ |  |  |  |  |  |  |  |  |  | F | S | S | D | S | S | X | M | W |  |  |
| LD (2) |  |  |  |  |  |  |  |  |  |  |  |  | F | S | S | F | D | X | M | W |

* The initial DADDI must pause until the LD instruction finishes and writes the value of R1 in the WB stage.
* The SD instruction also needs to wait until that DADDI finishes calculating the new value of R1 and completes its writeback (since there's no forwarding).
* The DSUB instruction must delay until the second DADDI finishes computing R2 and reaches the writeback phase.
* The BNEZ instruction depends on the result of DSUB, so it must wait until R4 is ready at the WB stage.
* The LD instruction for the next loop iteration is fetched only after the branch outcome is known and resolved as taken during the decode (ID) stage.

Since the loop runs 99 times (396 ÷ 4), covering iterations 0 through 98, each iteration starts at cycle number 1 + (i × 15). The final iteration takes 18 cycles to finish. Therefore, the total execution time for the full loop is (98 × 15) + 18 = 1488 cycles.

1. **Forwarding and branching predict not taken**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| LD | F | D | X | M | W |  |  |  |  |  |  |  |  |  |
| DADDI |  | F | D | S | X | M | W |  |  |  |  |  |  |  |
| SD |  |  | F | S | D | X | M | W |  |  |  |  |  |  |
| DADDI |  |  |  |  | F | D | X | M | W |  |  |  |  |  |
| DSUB |  |  |  |  |  | F | D | X | M | W |  |  |  |  |
| BNEZ |  |  |  |  |  |  | F | D | S | X | M | W |  |  |
| LD (2) |  |  |  |  |  |  |  | F | s | F | D | X | M | W |

* The initial ADDI still must wait for the LD instruction to reach the writeback (WB) stage to access the updated value of R1. However, the stall now occurs in the execute (EX) stage due to the interlocking mechanism, and the value is directly forwarded to the EX-stage.
* The SD instruction waits for the ADDI to finish calculating the new value of R1, which is then forwarded from the EX stage to the memory (MEM) stage.
* The DSUB instruction receives the value of R2 via forwarding from the second ADDI and can proceed without any delay.
* The BNEZ instruction also gets the R4 value from DSUB through forwarding, but it must hold off until DSUB reaches the EX stage.
* The LD instruction in the next loop iteration is fetched again after the branch is incorrectly predicted and resolved in the instruction decode (ID) stage.

Each iteration of the loop starts at cycle 1 + (i × 8), and the final iteration takes 11 cycles to finish. So, the entire loop runs in (98 × 9) + 12 = 894 cycles

1. **Simple instruction scheduling and branch delay slot.**

* **Improvement 1**: Relocate the second DADDI instruction into the delay slot following the load (LD) instruction.
* **Improvement 2**: Shift the second DSUB instruction upward so that it comes right after the relocated DADDI.
* **Improvement 3**: Place the SD instruction into the branch delay slot and modify its address to SD -4(R2), R1 to maintain correct data storage.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

* The first DADDI proceeds without delay because it's now spaced one cycle apart from the LD instruction.
* The BNEZ instruction also executes without stalling since it’s now offset by an additional cycle from the DSUB.
* The SD instruction has been positioned in the branch delay slot, ensuring that the slot is used effectively.
* The LD instruction for the next loop iteration is correctly fetched on time because the branch decision is resolved early enough.

Each loop iteration begins at cycle 1 + (i × 6), and the final one takes 10 cycles to complete. Therefore, the total cycle count for the entire loop is (98 × 6) + 10 = 598 cycles.