# Problem 1 Part A

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lw x1, 0(x2) | F | D | X | M | W | | | | | | | | | | | | | | | | |
| Lw x3, 0(x4) | | F | D | X | M | W | | | | | | | | | | | | | | | |
| Mul x1, x1, x6 | | | F | D | X | M | W | | | | | | | | | | | | | | |
| Mul x3, x3, x7 | | | | F | D | X | M | W | | | | | | | | | | | | | |
| Add x8, x1, x3 | | | | | F | D | X | M | W | | | | | | | | | | | | |
| Add x9, x9, x8 | | | | | | F | D | X | M | W | | | | | | | | | | | |
| Addi x2, x2, 4 | | | | | | | F | D | X | M | W | | | | | | | | | | |
| Addi x4, x4, 4 | | | | | | | | F | D | X | M | W | | | | | | | | | |
| Addi x10, x10, -1 | | | | | | | | | F | D | X | M | W | | | | | | | | |
| Bne x10, x0, loop | | | | | | | | | | F | D | X | M | W | | | | | | | |
| OP A | | | | | | | | | | | F | D | - | - | - | | | | | | |
| OP B | | | | | | | | | | | | F | - | - | - | - | | | | | |
| LOOP: | | | | | | | | | | | | | F | D | X | M | W | | | | |

**CPI = 12 cycles / 10 instructions = 1.20 cycles**

**IPC = 10 instructions / 12 cycles = 0.83 instructions**

# Problem 1 Part B

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw x1, 0(x2) | F | D | B0 | B1 | W | | | | | | | | | | | | | |
| lw x3, 0(x4) | F | D | D | B0 | B1 | W | | | | | | | | | | | | |
| mul x1, x1, x6 | | F | F | D | A0 | A1 | W | | | | | | | | | | | |
| mul x3, x3, x7 | | F | F | D | D | A0 | A1 | W | | | | | | | | | | |
| add x8, x1, x3 | | | | F | F | D | A0 | A1 | W | | | | | | | | | |
| add x9, x9, x8 | | | | F | F | D | D | A0 | A1 | W | | | | | | | | |
| addi x2, x2, 4 | | | | | | F | F | D | A0 | A1 | W | | | | | | | |
| addi x4, x4, 4 | | | | | | F | F | D | B0 | B1 | W | | | | | | | |
| addi x10, x10, -1 | | | | | | | | F | D | A0 | A1 | W | | | | | | |
| bne x10, x0, loop | | | | | | | | F | D | D | A0 | A1 | W | | | | | |
| OP A | | | | | | | | | F | F | D | - | - | - | | | | |
| OP B | | | | | | | | | F | F | D | - | - | - | | | | |
| OP C | | | | | | | | | | | F | - | - | - | - | | | |
| OP D | | | | | | | | | | | F | - | - | - | - | | | |
| loop: | | | | | | | | | | | | F | D | B0 | B1 | W | | |

**CPI = 11 cycles / 10 instructions = 1.10 cycles**
**IPC = 10 instructions / 11 cycles = 0.91 instructions**

# Problem 1 Part C

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw x1, 0(x2) | F | D | B0 | B1 | W | | | | | | | | | | | | | |
| lw x3, 0(x4) | F | D | D | B0 | B1 | W | | | | | | | | | | | | |
| mul x1, x1, x6 | | F | F | D | A0 | A1 | W | | | | | | | | | | | |
| addi x2, x2, 4 | | F | F | D | B0 | B1 | W | | | | | | | | | | | |
| mul x3, x3, x7 | | | | F | D | A0 | A1 | W | | | | | | | | | | |
| addi x4, x4, 4 | | | | F | D | B0 | B1 | W | | | | | | | | | | |
| addi x10, x10, -1 | | | | | F | D | A0 | A1 | W | | | | | | | | | |
| add x8, x1, x3 | | | | | F | D | B0 | B1 | W | | | | | | | | | |
| add x9, x9, x8 | | | | | | F | D | B0 | B1 | W | | | | | | | | |
| bne x10, x0, loop | | | | | | F | D | A0 | A1 | W | | | | | | | | |
| OP A | | | | | | | F | D | - | - | - | | | | | | | |
| OP B | | | | | | | F | D | - | - | - | | | | | | | |
| OP C | | | | | | | | F | - | - | - | - | | | | | | |
| OP D | | | | | | | | F | - | - | - | - | | | | | | |
| loop: | | | | | | | | | F | D | B0 | B1 | W | | | | | |

**CPI = 8 cycles / 10 instructions = 0.80 cycles**

**IPC = 10 instructions / 8 cycles = 1.25 instructions**

# Problem 1 Part D

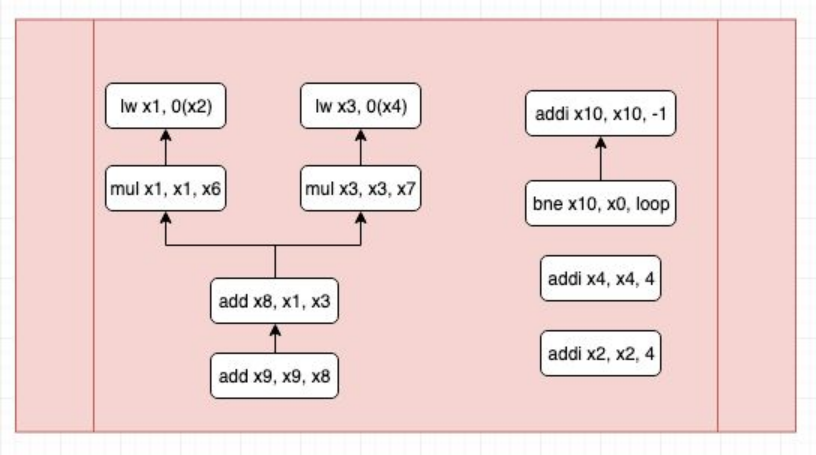
| Lw x1, 0(x2) | F | D | B0 | B1 | W | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addi x2, x2, 4 | F | D | A0 | A1 | W | | | | | | | | | | | | | |
| Lw x3, 0(x4) | F | D | H0 | H1 | W | | | | | | | | | | | | | |
| Addi x4, x4, 4 | F | D | G0 | G1 | W | | | | | | | | | | | | | |
| Mul x1, x1, x6 | | F | D | D | A0 | A1 | W | | | | | | | | | | | |
| Mul x3, x3, x7 | | F | D | D | G0 | G1 | W | | | | | | | | | | | |
| Add x8, x1, x3 | | F | D | D | D | B0 | B1 | W | | | | | | | | | | |
| Addi x10, x10, -1 | | F | D | H0 | H1 | W | | | | | | | | | | | | |
| Add x9, x9, x8 | | | F | F | F | D | H0 | H1 | W | | | | | | | | | |
| Bne x10, x0, loop | | | F | F | F | D | A0 | A1 | W | | | | | | | | | |
| OP A | | | F | F | F | D | B0 | - | - | | | | | | | | | |
| OP B | | | F | F | F | D | G0 | - | - | | | | | | | | | |
| OP C | | | | | | F | D | - | - | - | | | | | | | | |
| OP D | | | | | | F | D | - | - | - | | | | | | | | |
| OP E | | | | | | F | D | - | - | - | | | | | | | | |
| OP F | | | | | | F | D | - | - | - | | | | | | | | |
| OP G | | | | | | | F | - | - | - | - | | | | | | | |
| OP H | | | | | | | F | - | - | - | - | | | | | | | |
| OP I | | | | | | | F | - | - | - | - | | | | | | | |
| OP J | | | | | | | F | - | - | - | - | | | | | | | |
| loop: | | | | | | | | F | D | B0 | B1 | W | | | | | | |

**CPI = 7 cycles / 10 instructions = 0.70 cycles**

**IPC = 10 instructions / 7 cycles = 1.43 instructions**

# Problem 1 Part E

**1 iteration ILP = 10 / 4 = 2.5**

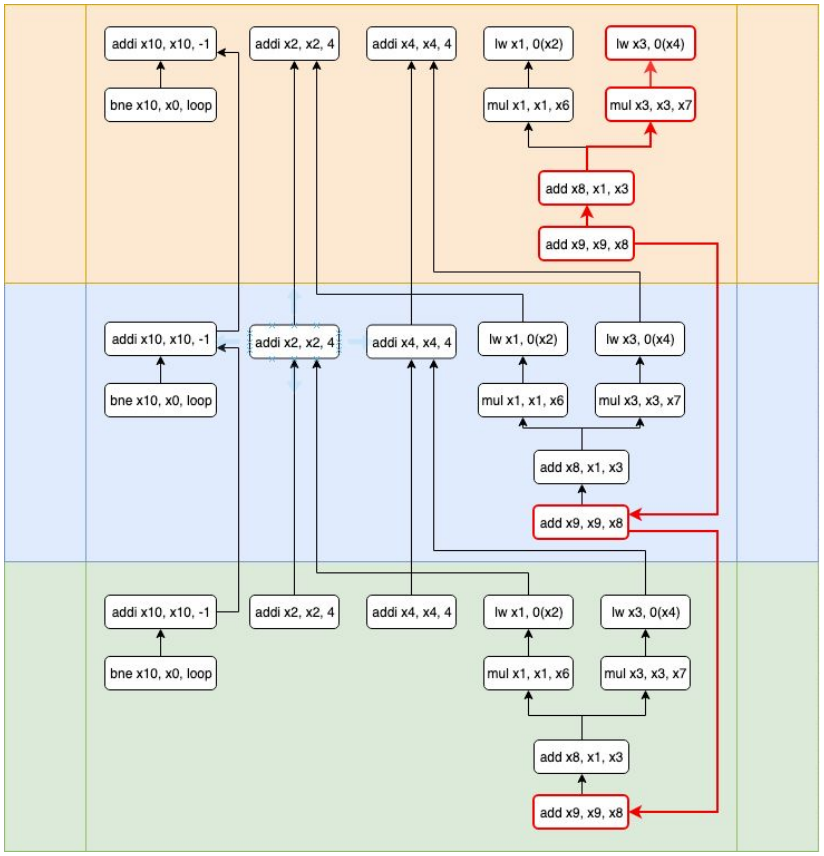



$$ILP = \frac{10N}{3+N}$$

**Discuss why the IPC achieved on the quad-issue TinyRV1 processor is less than the ideal ILP.**

IPC for quad-issue processor is much lower than the ideal ILP because as shown in the 3 iteration graph, there are cross dependencies among instructions in-between iterations. Registers such as x2 and x4 have multiple dependent nodes. The fact that the values x2 and x4 are available in the latter stages in the pipeline makes it even harder to parallelize the instructions. This means, we have to slip the instructions in the same fetch block, thus reducing the IPC. We cannot simply execute all instructions in parallel.

**3 iteration ILP = 30 / 6 = 5**



# Problem 2 Part A

**Un-optimized code**

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw x1, 0(x2) | F | D | I | L0 | L1 | W | C | | | | | | | | | | | | | | |
| mul x3, x1, x4 | F | D | I | I | I | Y0 | Y1 | Y2 | Y3 | W | C | | | | | | | | | | |
| sw x3, 0(x5) | | F | D | D | D | I | I | I | I | S | W | C | | | | | | | | | |
| addi x2, x2, 4 | | F | D | D | D | I | I | I | I | A | W | C | | | | | | | | | |
| addi x5, x5, 4 | | | F | F | F | D | D | D | D | I | A | W | C | | | | | | | | |
| addi x6, x6, -1 | | | F | F | F | D | D | D | D | I | B | W | C | | | | | | | | |
| bne x6, x0, loop | | | | | | F | F | F | F | D | I | A | W | C | | | | | | | |
| - | | | | | | F | F | F | F | D | - | - | - | - | | | | | | | |
| loop: | | | | | | | | | | F | D | I | L0 | L1 | W | C | | | | | |

**Total cycles = 6 initial + 64 iterations *(9 cycles/iteration) = 582 cycles**

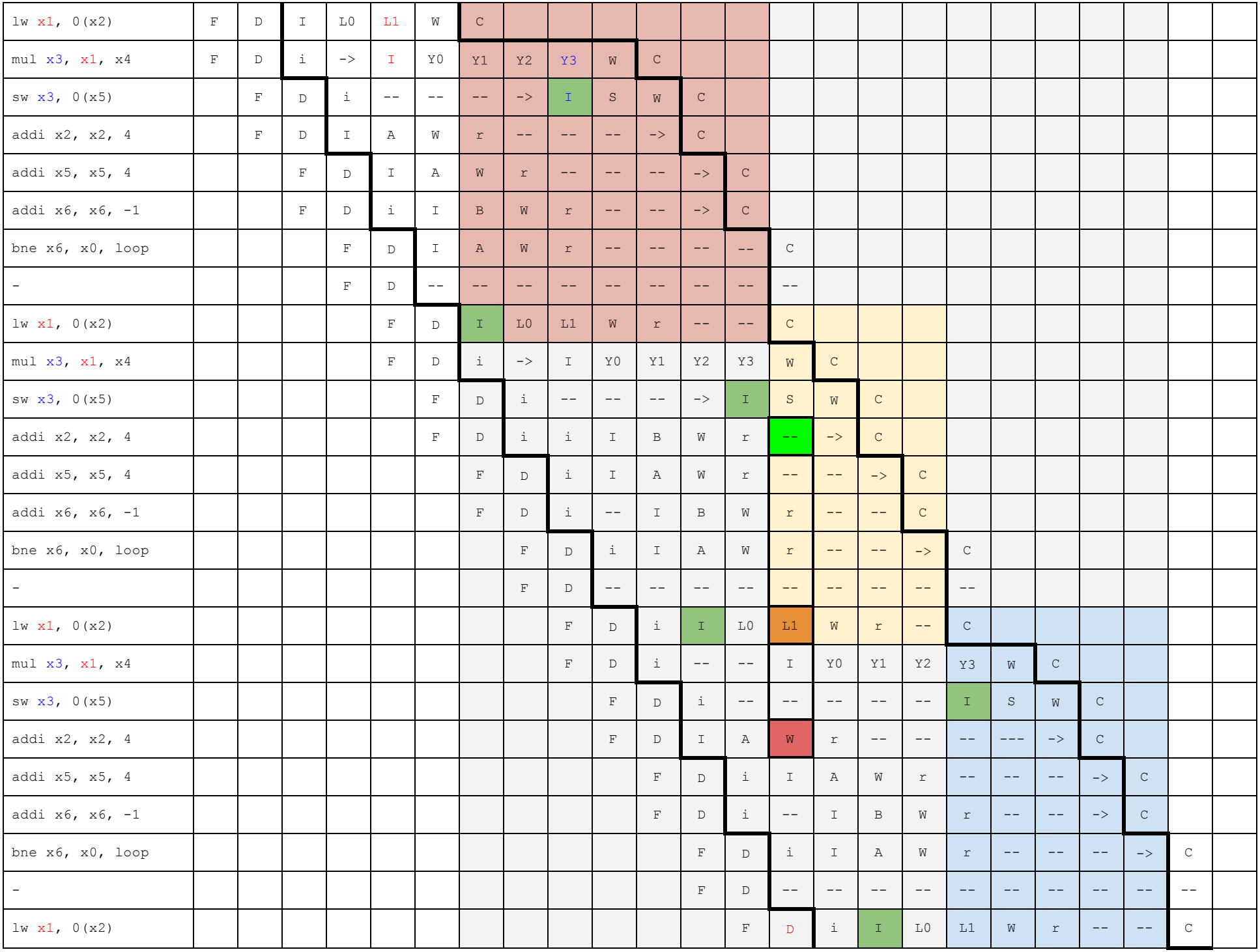
**Optimized code**

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw x1, 0(x2) | F | D | I | L0 | L1 | W | C | | | | | | | | | | | | | | |
| addi x2, x2, 4 | F | D | I | A | W | r | C | | | | | | | | | | | | | | |
| mul x3, x1, x4 | | F | D | I | I | Y0 | Y1 | Y2 | Y3 | W | C | | | | | | | | | | |
| addi x6, x6, -1 | | F | D | I | I | B | W | r | -- | -- | C | | | | | | | | | | |
| sw x3, 0(x5) | | | F | D | D | I | I | I | I | S | W | C | | | | | | | | | |
| addi x5, x5, 4 | | | F | D | D | I | I | I | I | A | W | C | | | | | | | | | |
| bne x6, x0, loop | | | | F | F | D | D | D | D | I | A | W | C | | | | | | | | |
| - | | | | F | F | D | D | D | D | - | - | - | - | | | | | | | | |
| loop: | | | | | | F | F | F | F | D | I | L0 | L1 | W | C | | | | | | |

**Total cycles = 6 initial + 64 iterations *(8 cycles/iteration) = 518 cycles**

# Problem 2 Part B

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw x1, 0(x2) | F | D | I | L0 | L1 | W | C | | | | | | | | | | | | | | | | | | |
| mul x3, x1, x4 | F | D | i | -> | I | Y0 | Y1 | Y2 | Y3 | W | C | | | | | | | | | | | | | | |
| sw x3, 0(x5) | | F | D | i | -- | -- | -- | -> | I | S | W | C | | | | | | | | | | | | | |
| addi x2, x2, 4 | | F | D | I | A | W | r | -- | -- | -- | -> | C | | | | | | | | | | | | | |
| addi x5, x5, 4 | | | F | D | I | A | W | r | -- | -- | -- | -> | C | | | | | | | | | | | | |
| addi x6, x6, -1 | | | F | D | i | I | B | W | r | -- | -- | -> | C | | | | | | | | | | | | |
| bne x6, x0, loop | | | | F | D | I | A | W | r | -- | -- | -- | -- | C | | | | | | | | | | | |
| - | | | | F | D | -- | -- | -- | -- | -- | -- | -- | -- | -- | | | | | | | | | | | |
| lw x1, 0(x2) | | | | | F | D | I | L0 | L1 | W | r | -- | -- | C | | | | | | | | | | | |
| mul x3, x1, x4 | | | | | F | D | i | -> | I | Y0 | Y1 | Y2 | Y3 | W | C | | | | | | | | | | |
| sw x3, 0(x5) | | | | | | F | D | i | -- | -- | -- | -> | I | S | W | C | | | | | | | | | |
| addi x2, x2, 4 | | | | | | F | D | i | i | I | B | W | r | -- | -> | C | | | | | | | | | |
| addi x5, x5, 4 | | | | | | | F | D | i | I | A | W | r | -- | -- | -> | C | | | | | | | | |
| addi x6, x6, -1 | | | | | | | F | D | i | -- | I | B | W | r | -- | -- | C | | | | | | | | |
| bne x6, x0, loop | | | | | | | | F | D | i | I | A | W | r | -- | -- | -> | C | | | | | | | |
| - | | | | | | | | F | D | -- | -- | -- | -- | -- | -- | -- | -- | -- | | | | | | | |
| lw x1, 0(x2) | | | | | | | | | F | D | i | I | L0 | L1 | W | r | -- | C | | | | | | | |
| mul x3, x1, x4 | | | | | | | | | F | D | i | -- | -- | I | Y0 | Y1 | Y2 | Y3 | W | C | | | | | |
| sw x3, 0(x5) | | | | | | | | | | F | D | i | -- | -- | -- | -- | -- | -- | I | S | W | C | | | |
| addi x2, x2, 4 | | | | | | | | | | F | D | I | A | W | r | -- | -- | -- | --- | -> | C | | | | |
| addi x5, x5, 4 | | | | | | | | | | | F | D | i | I | A | W | r | -- | -- | -- | -> | C | | | |
| addi x6, x6, -1 | | | | | | | | | | | F | D | i | -- | I | B | W | r | -- | -- | -> | C | | | |
| bne x6, x0, loop | | | | | | | | | | | | F | D | i | I | A | W | r | -- | -- | -- | -> | C | | |
| - | | | | | | | | | | | | F | D | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | | |
| lw x1, 0(x2) | | | | | | | | | | | | | F | D | i | I | L0 | L1 | W | r | -- | -- | C | | |



**Total cycles = 6 initial + (1 iteration * 7 cycles/iteration) +**
**+ (31 iterations * 9 cycles/iteration) +**
**+ (1 iteration * 4 cycles/iteration) = 582 cycles**

# Problem 2 Part C

What kind of branch predictor would we need to redirect the control flow in the fetch stage?

We would need a branch predictor that decided the branch logic in early D stage so that we fetched the correct instruction in the same clock cycle.

Identify a situation where an instruction renames its architectural destination register, while another instruction is still in-flight that writes the same register.

This situation is highlighted and bordered in the pipeline diagram above. This situation arises when the load instruction has to **lw** a value from address **x2** in the **L1** stage. The value stored in **x2** must come from the value calculated in the previous iteration. This means that in the previous iteration **x2** was assigned a physical register, and is currently in the reorder buffer. Since, the latter **addi** (which calculates the new value of **x2**) issues at the same time as **lw**, **addi** gets to **W** at the same time as **lw** gets to **L1**. This means that if the register renaming did not take place in the **I** stage, new value of **x2** would still point to the old physical register and update its' value before **lw** instruction has the chance to read it. This might result in the **lw** reading the wrong **x2** address.

Identify a situation where a load is able to issue before an earlier store in program order due to out-of-order load/store issue.

Load instruction is any iteration after the 1st iteration seems to be issued before the store instruction of the previous iteration. The next array value is loaded before committing the previously calculated array value is stored away. This is highlighted with a green in the pipeline diagram.

Are there any situations in this loop where we would need to replay the load that issued speculatively?

Since **lw** depends on the value of **x2**, in each iteration **lw** can safely issue because the **x2** value from previous iteration is available in the reorder buffer given that register renaming is properly executed.

How does the performance of the previous two parts compare?

The total number of cycles has **decreased by ~2 times**, meaning the performance has doubled in OOO processor.

Is the out-of-order processor able to achieve higher performance than the in-order processor?

Yes.

Would optimizing the code significantly help for the in-order processor? Why or why not?

No, because even with optimized code, the throughput of in-order processor is lower than the throughput of the out-of-order one. This is evident in the narrow gap between the dark lines in the pipeline diagram in part A compared to part B. The total cycles with optimized code is 518 cycles (64 cycles less than unoptimized), which is only about 11% reduction as shown in part A.