

- c. Hand-unroll two iterations of the loop in your reordered code from Exercise 2.5. What speedup did you obtain? (For this exercise, just color the $N + 1$ iteration's instructions green to distinguish them from the N th iteration's; if you were actually unrolling the loop you would have to reassign registers to prevent collisions between the iterations.)
- 2.7 [15] <2.1> Computers spend most of their time in loops, so multiple loop iterations are great places to speculatively find more work to keep CPU resources busy. Nothing is ever easy, though; the compiler emitted only one copy of that loop's code, so even though multiple iterations are handling distinct data, they will appear to use the same registers. To keep register usages multiple iterations from colliding, we rename their registers. Figure 2.36 shows example code that we would like our hardware to rename.

A compiler could have simply unrolled the loop and used different registers to avoid conflicts, but if we expect our hardware to unroll the loop, it must also do the register renaming. How? Assume your hardware has a pool of temporary registers (call them T registers, and assume there are 64 of them, T0 through T63) that it can substitute for those registers designated by the compiler. This rename hardware is indexed by the source register designation, and the value in the table is the T register of the last destination that targeted that register. (Think of these table values as producers, and the src registers are the consumers; it doesn't much matter where the producer puts its result as long as its consumers can find it.) Consider the code sequence in Figure 2.36. Every time you see a destination register in the code, substitute the next available T, beginning with T9. Then update all the src registers accordingly, so that true data dependences are maintained. Show the resulting code. (*Hint*: See Figure 2.37.)

```

Loop: LD    F2,0(Rx)
I0:  MULTD F5,F0,F2
I1:  DIVD  F8,F0,F2
I2:  LD    F4,0(Ry)
I3:  ADDD  F6,F0,F4
I4:  ADDD  F10,F8,F2
I5:  SD    F4,0(Ry)

```

Figure 2.36 Sample code for register renaming practice.

```

I0:  LD    T9,0(Rx)
I1:  MULTD T10,F0,T9
. . .

```

Figure 2.37 Expected output of register renaming.