Based on the MIPS code above:

1. How many nops do you need and where to remove the hazards in the code <with no forwarding>?

   |   | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 |
---|---|---|---|---|---|---|---|---|---|---|---|---|
lw | IF | ID | EX | MEM | WB |
   | nop | --- | --- | --- | --- |
   | nop | --- | --- | --- | --- |
addi | IF | ID | EX | MEM | WB |
   | nop | --- | --- | --- | --- |
   | nop | --- | --- | --- | --- |
sw | --- | --- | --- | --- | --- |

   a. We need 2 nops between lw and addi to make sure we WB to t2 before reading its value in the ID stage of addi.

   b. We need 2 more nops between addi and sw to make sure that we WB to t2 before reading its value in ID stage of sw.

Note: If we assume that we have the same memory for both instruction and data, we have a structural hazard here between lw and addi in C4, when they are both trying to read the memory. We can take care of that by replacing a nop between lw and addi with a stall.

   |   | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 |
---|---|---|---|---|---|---|---|---|---|---|---|---|
lw | IF | ID | EX | MEM | WB |
   | nop | --- | --- | --- | --- |
addi | IF | stall | ID | EX | MEM | WB |
   | nop | --- | --- | --- | --- |
   | nop | --- | --- | --- | --- |
sw | --- | --- | --- | --- | --- |

2. How many nops do you need and where to remove the hazards in the code <with ALU-ALU only forwarding>?

   a. We can’t do any ALU-ALU forwarding between lw and addi here because our t2 is only ready after the MEM stage in lw. So we still need 2 nops between lw and addi.

   b. For the second hazard ALU-ALU forwarding works because the data is ready at the end of the EXE phase of the addi and sw requires this data at the start of its ID phase. In the diagram below you can see that this is possible.
3. How many nops do you need and where to remove the hazards in the code <with full forwarding>?

   a. Now, MEM of lw can forward t2 from C4 to EX of addi in C5, but we still 1 more nop for t2 to be ready i.e. for the lw to finish the loading t2 from memory [*]
   b. EX of addi can forward t2 from C5 to EX of sw in C6 directly, so we don’t need any nops [**]

4. Assuming the following latencies for each pipeline stages, What is the speedup achieved by implemented the pipelined [full-forwarding] CPU over non-pipelined?

   IF: 200ps  
   ID: 120ps 
   EX: 150ps 
   MEM: 190ps 
   WB: 100ps

   With full forwarding pipeline we have, 8 cycles, total time 
   = 8 cycles * max. pipeline stage time = 8*200 = 1600ps

   For a single cycle processor,

   Time for 1 instruction = IF + ID+ EX + MEM + WB = 200 + 120 + 150 + 190 + 100 = 760ps

   Total time = number of instructions * time per instruction = 3 * 760 = 2280ps

   Speedup, pipeline/non-pipeline = time for non-pipeline/time for pipeline = 2280/1600 = 1.425

5. Is the single cycle CPU faster than the pipelined CPU without forwarding?

   Total execution time for pipeline with no forwarding = 200*11cycles = 2200 
   Which is less time than the single cycle CPU, so yes, pipelined CPU with 2 nops between instructions still outperforms the single cycle CPU.