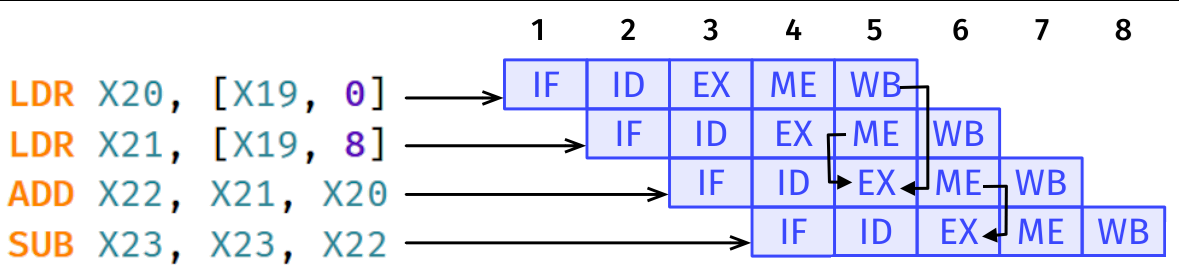
Daniel Detore

I pledge my honor that I have abided by the Stevens Honor System.

1. 1. All branching instructions would break if the Br wire were stuck at 0. No program would ever branch and the CPU would read it straight through. The broken instructions are B, BL, CBZ, CBNZ, and RET.
   2. Arithmetic and logic instructions that use immediate values would break and always erroneously read garbage from a register if ALUsrc were stuck at 0. This breaks ADD, ADDS, SUB, SUBS, ORR, and CMP but only in the syntax which reads one register and one immediate.
   3. If RegWrite is stuck at 0 then any instruction that writes to a register breaks, as they will not write to a register. This includes all syntaxes of ADD, ADDS, SUB, SUBS, AND, ANDS, ORR, LDR and STR.
2. There would need to be a temporary register that holds the value of Rd so that its original value does not get overwritten. Then the value of Rn can be moved into Rd and the temporary register can move its value into Rn. Alternatively, we could add a RegWrite2 wire so we can read and write from both Rd and Rn in one instruction.
3. 1. Without the improvement, the clock cycle time must be the sum of the latencies of all 5 stages. That is 200 ps + 250 ps + 150 ps + 300 ps + 200 ps = **1100 ps**.  
      Adding 200 ps latency to the ALU increases the latency in the EX stage from 300 ps to 500 ps. The new clock cycle is 200 ps + 250 ps + 150 ps + 500 ps + 200 ps = **1300 ps**.
   2. We run a program with *n* instructions on both CPUs. The CPU time on the old CPU is *n* \* 1 \* 1100 = 1100*n* ps. On the old CPU, the same program takes .80*n* \* 1 \* 1300 = 1040*n* ps. This gives us the speedup 1 - (1040*n* / 1100*n* ) = **0.05455**.
   3. 1 - (.8x / 1100) < 0 ⇒ x = 1375. The CPU can have as much as **1375 ps** of latency before the speedup becomes a slowdown.
4. 1. In a pipelined processor, the clock cycle would need to be max(200, 250, 150, 300, 200) = **300 ps**. Without the pipeline, the clock cycle must be 200 + 250 + 150 + 300 + 200 = **1100 ps**.
   2. The LDR instruction requires all 5 stages of execution. On a pipelined CPU, its latency would be 300 \* 5 = **1500 ps**. On a non-pipelined CPU, all 5 stages happen in one clock cycle, which is **1100 ps** long.
   3. I would split the ME stage into 2 parts because it's the longest. If the pipelined CPU can only be as fast as its slowest stage then logically we should shorten its longest stage. This makes the new latency max(200, 250, 150, 150, 150, 200) = **250 ps**.
5. The value of X20 is forwarded from the WB stage of instruction 1 to the EX stage of instruction 3.  
   The value of X21 is forwarded from the ME stage of instruction 2 to the EX stage of instruction 3.  
   The value of X22 is forwarded from the ME stage of instruction 3 to the EX stage of instruction 4.
6. 1. LDR X1, [X6, 8] //1

NOP

NOP

ADD X0, X1, X0 //2

NOP

NOP

STR X0, [X10, 4] //3

LDR X2, [X6, 12] //4

NOP

NOP

SUB X3, X0, X2 //5

NOP

NOP

STR X3, [X8, 24] //6

CBZ X2, 40 //7

* 1. LDR X1, [X6, 8] //1

LDR X2, [X6, 12] //4

ADD X0, X1, X0 //2

SUB X3, X0, X2 //5

STR X0, [X10, 4] //3

STR X3, [X8, 24] //6

CBZ X2, 40 //7