Cycle	Instruction Retired	Reason
1-7	NOP	The processor is starting to execute instructions and hasn't had any instructions reach the writeback phase yet (where they are retired). After rst is deasserted on cycle 3, instructions begin to execute.
8	lbi r0, 0	
9	lbi r5, 43	
10	lbi r6,43	
11	lbi r7, 43	
12	ld r1, r0, 0	
13	NOP	When the ld instruction was in the execute stage, the store instruction was in the decode stage. This means that we knew that Rs = R1 for the st instruction and Rd = R1 for the load instruction, a RAW dependency. We don't have the proper result available for R1, as this needs to be fetched from memory. As such, we stall the pipeline by injecting a NOP into the DE/EX stage so the result can be fetched from memory, and then we forward the value from the WB stage to the EX stage.
15	st r5, r1, 0	
15	ld r1, r0, 2	
16	NOP	Same reasoning as above, the store depends on the load operation (R1 RAW), and this result isn't available, so we stalled.
17	st r6, r1, 1	
18	ld r1, r0, 4	
19	NOP	Same reasoning as above, the store depends on the load operation (R1 RAW), and this result isn't available, so we stalled.
19	st r7, r1, 1	
20	NOP	In order to mitigate an issue with a store instruction following halt instructions (as a store could completely execute if immediately following a valid halt signal), I hold the PC when a halt occurs in the HDU. This in turn creates a NOP instruction from the hold. I tried to balance this one wasted cycle (which I could not find out a better way to do) by implementing forwarding instead of stalling for RAW.
21	halt	