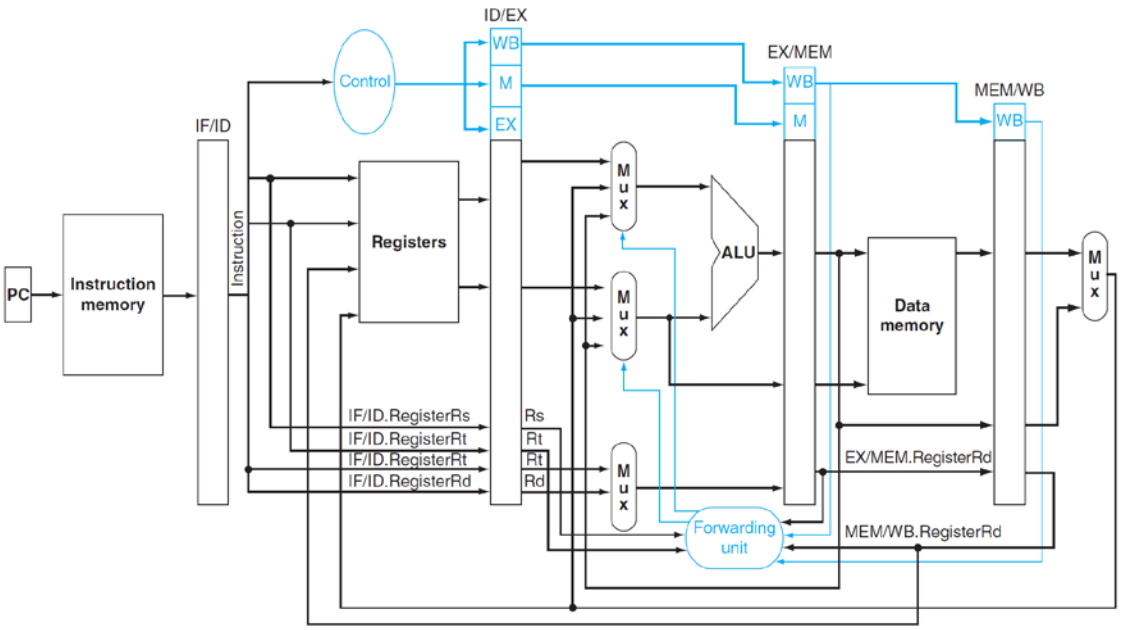
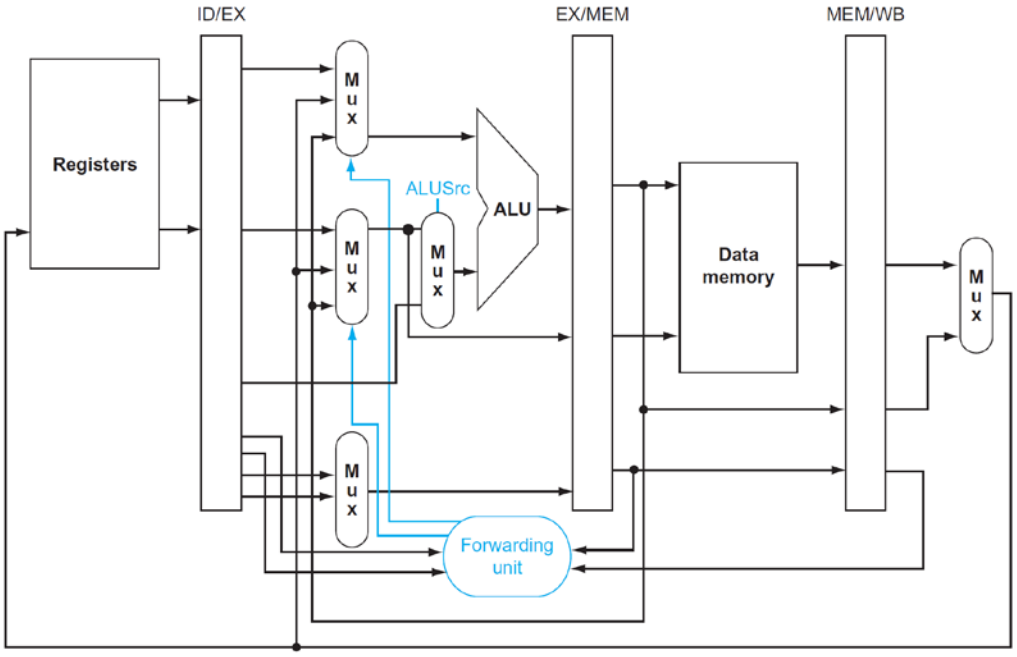
ECEC 500 – FUNDAMENTALS OF COMPUTER HARDWARE

Objective:

Data hazard to be overcome by implementing forwarding circuitry





Taking the result from the earliest point that it exists in any of the pipeline state registers and forward it to the functional units (e.g., the ALU) that need it that cycle.

For ALU functional unit: the inputs can come from any pipeline register rather than just from ID/EX by

- adding multiplexors to the inputs of the ALU
- connecting the Rd write data in EX/MEM or MEM/WB to either connecting the Rd write data in EX/MEM or MEM/WB to either (or both) of the EX's stage Rs and Rt ALU mux inputs.
- adding the proper control hardware to control the new muxes.

Other functional units may need similar forwarding logic (e.g., the DM)

- With forwarding can achieve a CPI of 1 even in the presence of data dependencies

Forwarding can also help with hazards when store instructions are dependent on other instructions. Since they use just one data value during the MEM stage, forwarding is easy. However, consider loads immediately followed by stores, useful when performing memory-to-memory copies in the MIPS architecture. Since copies are frequent, we need to add more forwarding hardware to make them run faster.

Initialization:

$s1 = 9; $s2 = 4; $s3 = 7; $s4 = 10

Program:

add $t1,$s1,$s4 -- $t1 <= 10 + 9 = 19

sub $t2,$s2,$s3 -- $t2 <= 4 – 7 = -3

add $s7,$t1,$t2 -- $s7 <= 19 + -3 = 16

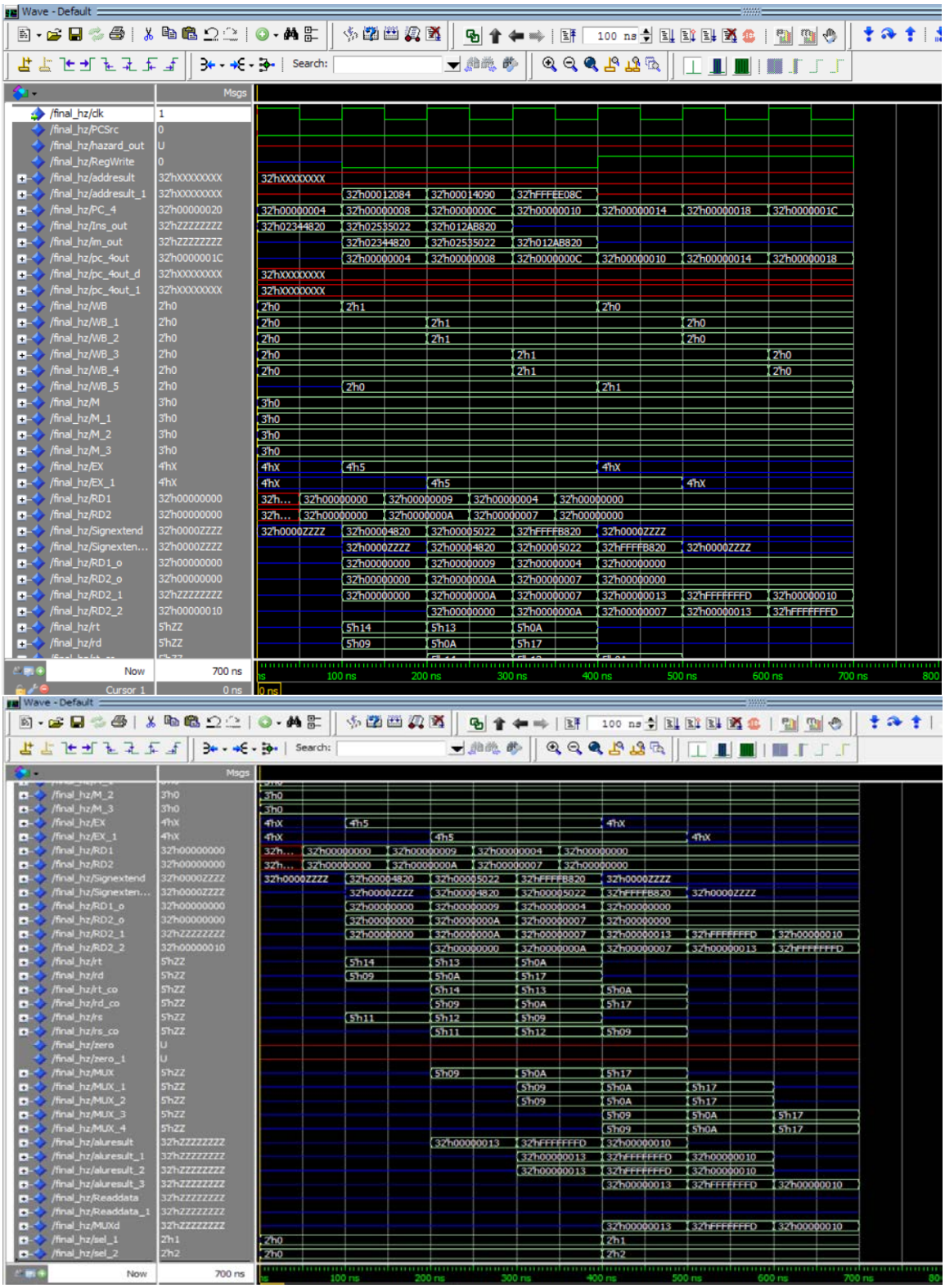
The first instruction adds the two values and stores it in the $t1 register.

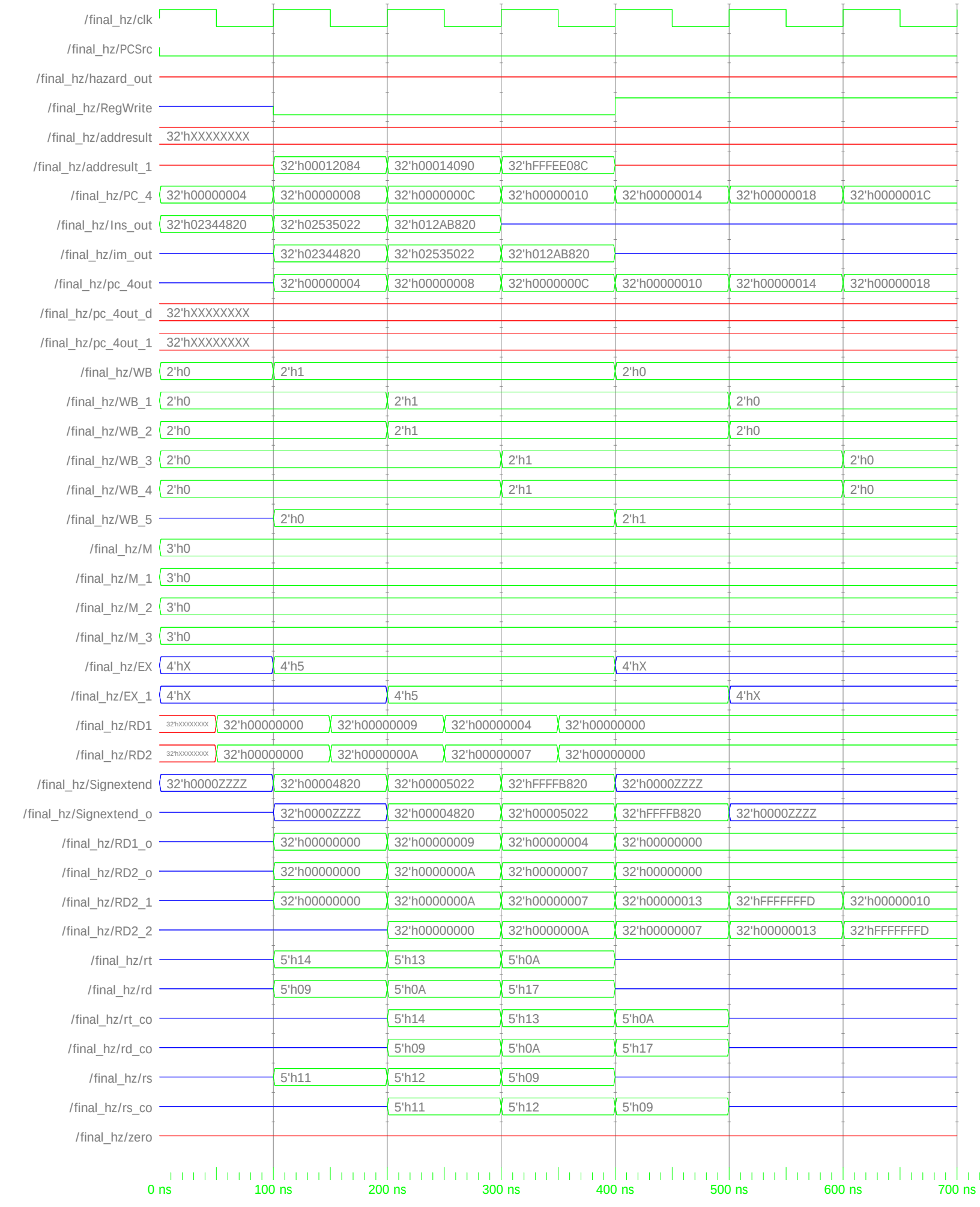
The second instruction subtracts the values and stores them in the $t2 register.

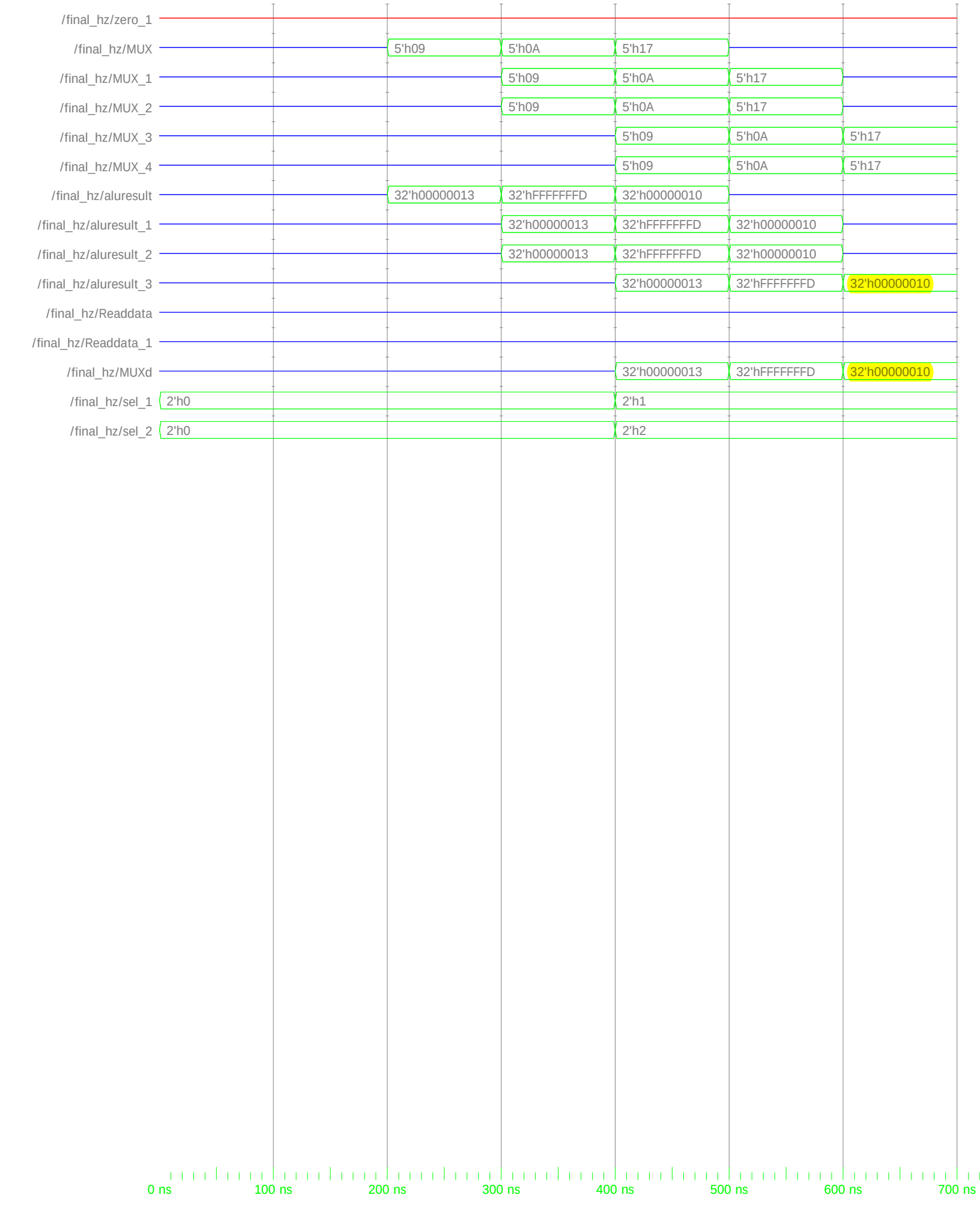
The third instruction adds and stores the values which are in $t1 and $t2.

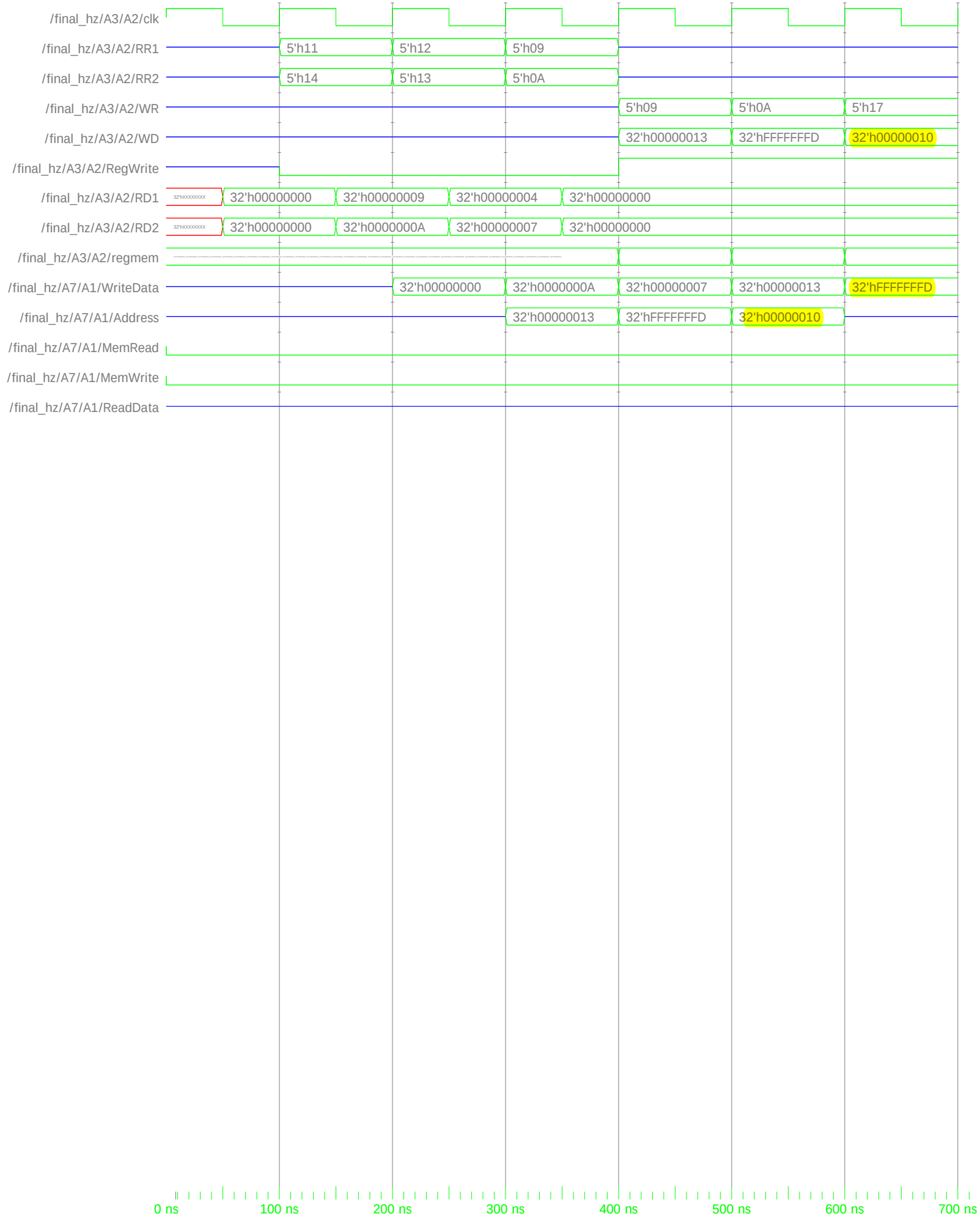
Since no stalls or delays are used, the first instruction executes in 5 cycles the second instruction finishes in the 6th cycle and the data is saved to the register $s7 of the third instruction in the 7th cycle.

ECEC 500 – FUNDAMENTALS OF COMPUTER HARDWARE




/final_hz/clk
/final_hz/PCSrc
/final_hz/hazard_out
/final_hz/RegWrite
/final_hz/addresult
32'hXXXXXXXX
/final_hz/addresult_1
32'h00012084
32'h00014090
32'hFFFEE08C
/final_hz/PC_4
32'h00000004
32'h00000008
32'h0000000C
32'h00000010
32'h00000014
32'h00000018
32'h0000001C
/final_hz/Ins_out
32'h02344820
32'h02535022
32'h012AB820
/final_hz/im_out
/final_hz/pc_4out
/final_hz/pc_4out_d
/final_hz/pc_4out_1
/final_hz/WB
2'h0
2'h1
/final_hz/WB_1
/final_hz/WB_2
/final_hz/WB_3
/final_hz/WB_4
/final_hz/WB_5
/final_hz/M
3'h0
/final_hz/M_1
/final_hz/M_2
/final_hz/M_3
/final_hz/EX
4'hX
4'h5
/final_hz/EX_1
/final_hz/RD1
32'hXXXXXXXX
32'h00000000
32'h00000009
32'h00000004
/final_hz/RD2
32'h0000000A
32'h00000007
/final_hz/Signextend
32'h0000ZZZZ
32'h00004820
32'h00005022
32'hFFFFB820
/final_hz/Signextend_o
/final_hz/RD1_o
/final_hz/RD2_o
/final_hz/RD2_1
/final_hz/RD2_2
/final_hz/rt
5'h14
5'h13
5'h0A
/final_hz/rd
5'h09
5'h17
/final_hz/rt_co
/final_hz/rd_co
/final_hz/rs
5'h11
5'h12
/final_hz/rs_co
/final_hz/zero
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns


/final_hz/zero_1
/final_hz/MUX
/final_hz/MUX_1
/final_hz/MUX_2
/final_hz/MUX_3
/final_hz/MUX_4
/final_hz/aluresult
/final_hz/aluresult_1
/final_hz/aluresult_2
/final_hz/aluresult_3
/final_hz/Readdata
/final_hz/Readdata_1
/final_hz/MUXd
/final_hz/sel_1
/final_hz/sel_2
5'h09
5'h0A
5'h17
32'h00000013
32'hFFFFFFFD
32'h00000010
32'hFFFFFFFD
2'h0
2'h1
2'h2
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns


/final_hz/A3/A2/clk
/final_hz/A3/A2/RR1
/final_hz/A3/A2/RR2
/final_hz/A3/A2/WR
/final_hz/A3/A2/WD
/final_hz/A3/A2/RegWrite
/final_hz/A3/A2/RD1
/final_hz/A3/A2/RD2
/final_hz/A3/A2/regmem
/final_hz/A7/A1/WriteData
/final_hz/A7/A1/Address
/final_hz/A7/A1/MemRead
/final_hz/A7/A1/MemWrite
/final_hz/A7/A1/ReadData
5'h11
5'h12
5'h09
5'h14
5'h13
5'h0A
5'h09
5'h0A
5'h17
32'h00000013
32'hFFFFFFFD
32'h00000010
32'h00000000
32'h00000009
32'h00000004
32'h00000000
32'h00000000
32'h0000000A
32'h00000007
32'h00000000
32'h00000000
32'h0000000A
32'h00000007
32'h00000013
32'hFFFFFFFD
32'h00000013
32'hFFFFFFFD
32'h00000010
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns