TASK 2: IMPLEMENTATION HELP FOR PIPELINED PROCESSOR AND FORWARDING UNIT

https://edaplayground.com/x/kS_U

APPROACH AND WORKING

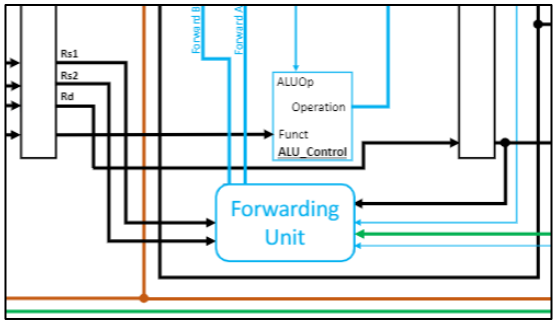
Here again we have used the modules of lab 11 but few other modules have also been compiled so that pipeline can work. The modules in this processor are:

- Immediate Generator
- MUX_64
- Instruction Parser
- Register File
- ALU_64
- Instruction Memory
- Data Memory
- PC Counter
- Adder
- Control Unit
- ALU Control
- IF/ID
- ID/EX
- Memory WB
- EX/Memory
- MUX 3 by 1
- Forwarding Unit

FOLLOWING ARE SOME OF THE MODIFICATIONS WE DID:

1) IF/ID, ID/EX, MEM/WB and EX/MEM modules were created. These are the registers that stores that values and forwards them to the next stage. They have equal number of inputs and outputs. Each of these had reset, clock and following other inputs.
    - **IF/ID: Input:** Instruction, PC out
    - **ID/EX: Inputs:** IF/ID Instruction, rs1, rs2, rd, immediate, read data 1, read data 2, PC, ALU operation, Branch, Memory Read, Memory to Reg, Memory Write, ALUSrc, Reg Write.
    - **MEM/WB: inputs:** EX/Mem Memory to Reg, EX/Memory Reg Write, Read Data, EX/Mem ALU result, EX/Mem rd,
    - **EX/Mem: Inputs:** ID/EX Branch, ID/EX Memory Read, ID/EX Memory to Reg, ID/EX Memory Write, ID/EX Reg Write, Adder, Zero, ALU Result, Forward_B MUX, ID/EX rd

2) In the previous labs we were using 2 by 1 MUX but here we wanted 3 by 1 MUX which takes 3 64 bits values and 1 2 bits selection line and give 1 64-bit output.
3) The last and the most important modification here is the Forwarding Unit. This module checks if the destination register in EX/MEM stage is equal to source register 1 or 2 in ID/EX stage. If they are equal it forwards the value directly to the next stage.



1a. EX/MEM.RegisterRd = ID/EX.RegisterRs1
1b. EX/MEM.RegisterRd = ID/EX.RegisterRs2
2a. MEM/WB.RegisterRd = ID/EX.RegisterRs1
2b. MEM/WB.RegisterRd = ID/EX.RegisterRs2

Figure a: book page 296

```
if   (MEM/WB.RegWrite
and  (MEM/WB.RegisterRd ≠ 0)
and  not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
        and (EX/MEM.RegisterRd = ID/EX.RegisterRs1))
and  (MEM/WB.RegisterRd = ID/EX.RegisterRs1)) ForwardA = 01

if   (MEM/WB.RegWrite
and  (MEM/WB.RegisterRd ≠ 0)
and  not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
        and (EX/MEM.RegisterRd = ID/EX.RegisterRs2))
and  (MEM/WB.RegisterRd = ID/EX.RegisterRs2)) ForwardB = 01
```

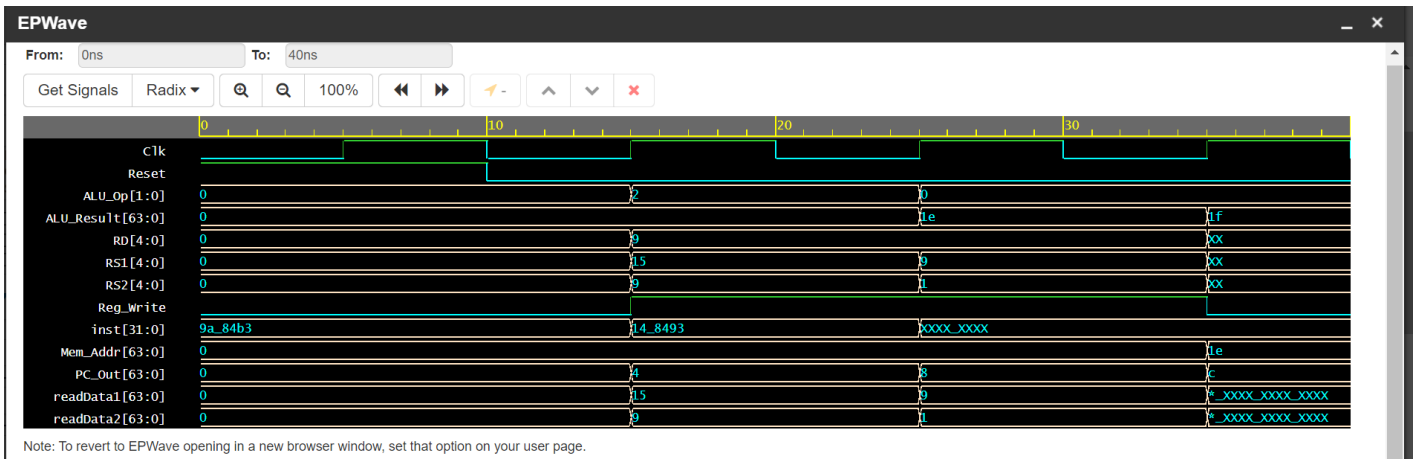
In order to check the working of this Pipeline processor we have fed following two instructions in the Instruction Memory:

Add x9, x21, x9

Addi x9, x9, 1

Here the value of x9 updates in the first instruction and then that updated value should be used in this next instruction, which will show if forwarding is working or not

RESULTS AND OUTPUTS



Here we can see that for instruction 1 the rd is 9(x9), and rs1, rs2 are 15(21 in decimal), 9 respectively. Also the ALUOp is 2 as it is an add instruction. The value x9 here is updated to 1e as the ALU result shows.

For instruction 2 we have rd = x9, rs1 = x9 and Immediate = 1. It should be 30 + 1 = 31 = 1F.

We can observe that updated value of x9 is used for instruction 2 and ALU result gives 1F.

Hence forwarding is working fine.