Instructions to update the Codasip files:

- For Phase 6, you will be using the completed project from Phase 5. If you have not completed Phase 5, work with your instructional staff to complete the phase.
- Import the hazard_detection_test project using the following git clone command
  - git clone https://github.com/CompOrg-RISCV/hazard_detection_test.git

- Changes to **the ca_defines.hcodal** file:
  - In the ID section create a new enum (rff_fwd) to control the data forwarding into the decode (ID) stage. You will need two values for register file forwarding (RFF): RFF_REG (register file output) and RFF_WB (MEM/WB pipeline latch value).
  - In the EX section create a new enum (dhf_fwd) to control the data forwarding in the execute (EX) stage. You will need three values for the data hazard forwarding (DHF): DHF_REG (register file output), DHF_ME (EX/MEM pipeline latch value), DHF_WB (MEM/WB pipeline latch value).
  - At the top of the file where the other widths definitions are, create the **two** new widths that can be composed from the two enums you just created (i.e. #define RFF_W bitsizeof(enum rff_fwd))

- Changes to the **ca_resources.codal** file:
  - In the ID section create the two new select variables (signals) for the two muxes that will control the forwarding of the data from the register file or the MEM/WB pipeline register. You need one per source operand (s_id_fwd1 an s_id_fwd2). Assign the corresponding variables widths.
  - In the ID section create the two variables your will need for the output of the two muxes. You will need to create two new variables (s_id_src1_val and s_id_src2_val) to hold either the forwarded value or the register file output value. Assign the corresponding variable widths.
  - In the ID section add two new pipeline registers to hold the names of the input operands (r_idex_rs1 and r_idex_rs2) to be able to implement the hazard detection logic. Assign the corresponding variable widths.
  - In the EX section create two new mux select variables to control the data forwarding for the two ALU input operands (s_ex_fwd1, s_ex_fwd2). Assign the corresponding variable widths.
  - In the EX section create two new variables to hold the output of the two muxes (s_ex_fwd1_val and s_ex_fwd2_val). These two variables will be the new inputs to the ALU in the ca_pipe3_ex.codal file. Assign the corresponding variable widths.
- Changes to the **ca_pipe2_id.codal** file:
  - After reading the register file into the corresponding signals, add the hazard logic detection to identify when the destination register that will be written in the writeback stage in this cycle is the same value that will be read from the register

file for the both input operands. The code below is just the beginning of the logic. Complete the rest of the code on your own.

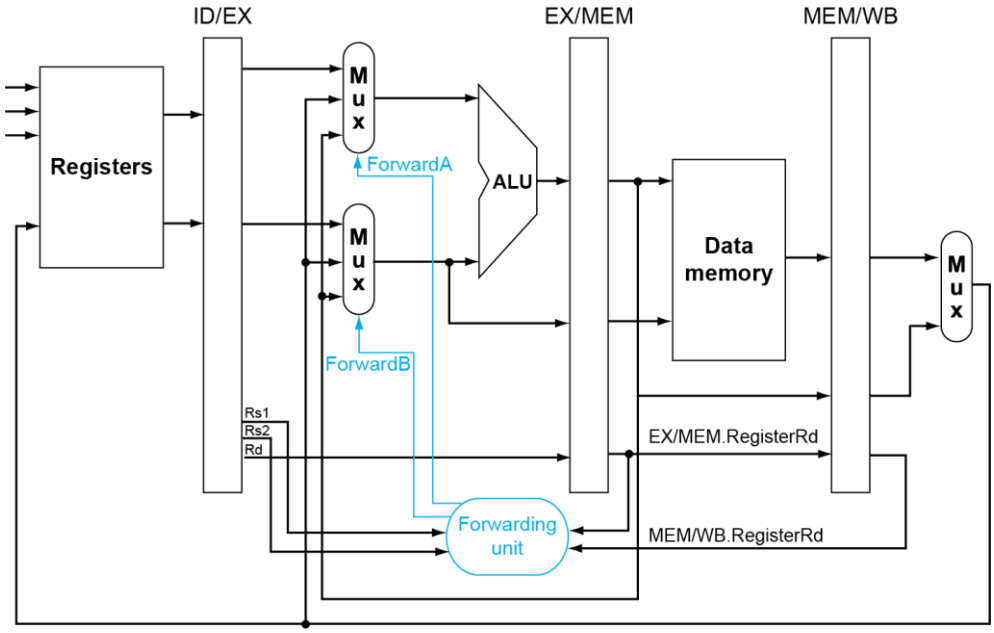
```
If ((r_mewb_rd == s_id_rs1) && (r_mewb_rd !=0) && r_mewb_regwrite == true)){
        (assign the corresponding value for the select variable for the mux)
}
If((r_mewb_rd …) …
```

- Create the 2 switch statements for the 2 muxes that will control the flow of the values. You should have already created all the variables needed in the ca_resources.codal file.
- Update the pipeline register assignments as needed. You will have to add two need ones and you may need to update two others.

- Changes to the **ca_pipe3_ex.codal** file:
  - At the beginning of the execute stage add the hazard detection logic and assign the mux select variables accordingly. You will need to do this for both r_idex_rs1 and r_idex_rs2. The code below is the beginning of the code you will have to write. Complete the rest of the code on your own.

```
If ((r_exme_rd == r_id_rs1) && (r_exme_rd !=0) && r_exme_regwrite == true)){
        (assign the select valriable to choose the value from the EX/MEM pipeline register)
}
else If((r_mem_rd …) …
```

  - Before calling the alu_operate function and before the muxes that select the different options for the alu input operands, update the register file values to the forwarded values as needed. To do this you will need to add the 2 switch statements to model the muxes that will control the flow of data (as shown in the schematics below). You should have already created the variables and enums necessary to complete this step and you should have assigned the select variables in the last buller point.
  - Update the variable that is assigned to the ouptut of the muxes that select between the multiple options for the alu input operands to the correspoding new variables (these should be the s_ex_fwd1_val instead of r_id_rf_src1 and s_ex_fwd2_val instead of r_id_rf_src2).


ID/EX
EX/MEM
MEM/WB
Registers
M
u
x
ForwardA
ALU
Data
memory
ForwardB
Rs1
Rs2
Rd
EX/MEM.RegisterRd
Forwarding
unit
MEM/WB.RegisterRd