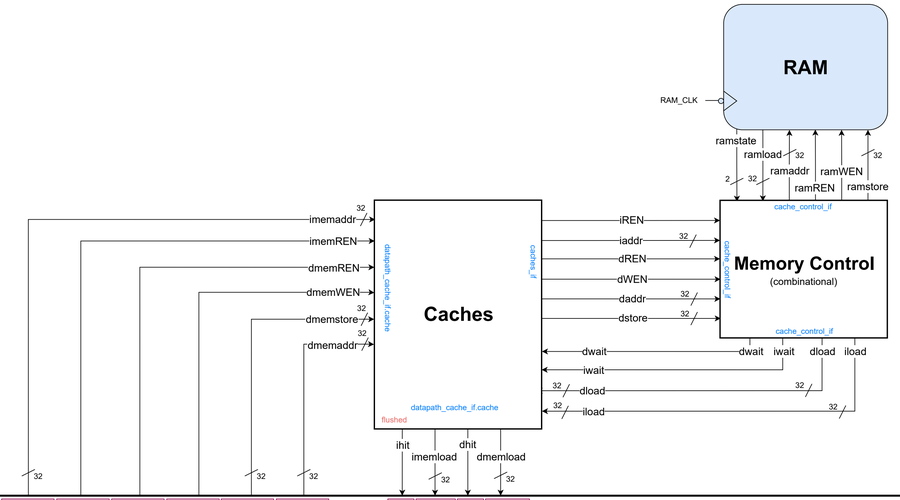
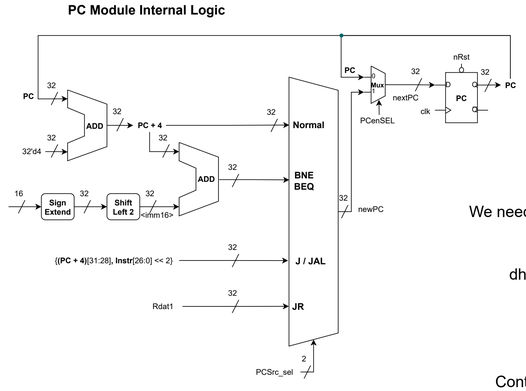
Potential hazards:
(1) PC+4 during a jal written in writeback, but used earlier by another instruction, like in the example hazard code in the `***asmFiles/hazards***` folder on GitHub

(2) Branch gets taken in MEM stage, but Jump in ID stage seen first
BEQ $0, $0, <label1> // should branch here
J <label2> // should not go here

(3) Encounter J instruction and HALT immediately after in pipe: cannot halt the PC



Very important to only allow the program counter to update when ihit is high since that is when we know the instruction being read will update! (*PCenSEL*)



JAL followed by JR requires forwarding the PC + 4 value

Branch in MEM while a jump is in Decode could cause conflicts in PCSrc determination

We need a stall signal which will cause each latch to retain its current output value to wait out a memory access

if ihit is high at the end of a cycle, stall = 0. else stall = 1
dhit is no longer needed? --> will need more complex stall logic in future

Implement logic to flush pipeline on branch incorrect guess --> in the future only, for now we shall ASSUME branch not taken and assembly code will insert NOPs for when the branch will be taken

Control unit update: RegDst is not a mux select anymore. It is the result of this mux's selection

Insert Nop after j type instruction as PC address can't be known until decode (1 cycle bubble like in correctly predicted ***branch taken***)