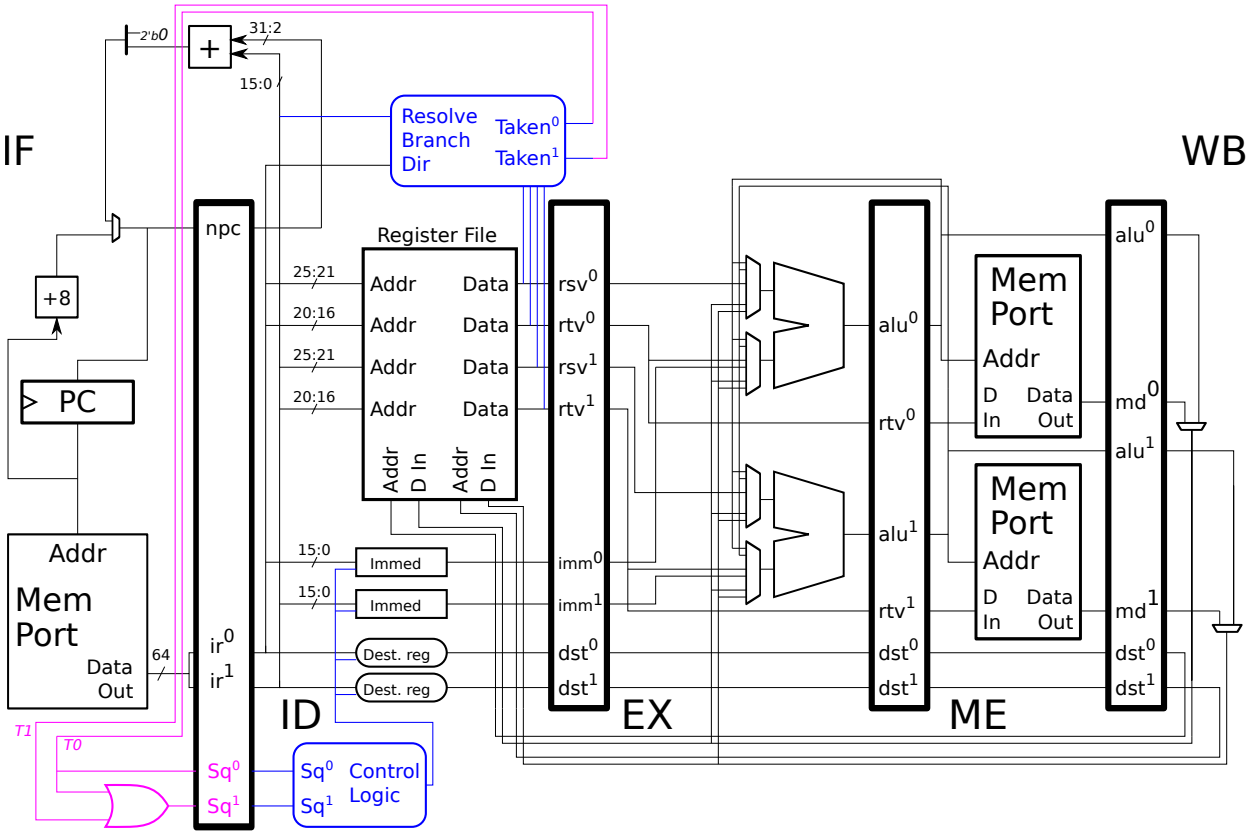
Problem 1: The following problem is an enhanced version of 2018 Final Exam Problem 1 (c). Appearing below is our 2-way superscalar MIPS with ID-stage hardware to determine branch direction (near the top in blue) and ID-stage hardware to squash instructions (near the bottom in blue). The Inkscape SVG source for this image can be found at `https://www.ece.lsu.edu/ee4720/2019/hw08-ss.svg`.

There are two outputs of the branch direction hardware logic, indicating whether the respective ID-stage slot has a taken branch. For example, if `Taken0` is `1` then there is a branch in slot 0 and that branch is taken. Of course, assume that this logic is correct.

There is a squash logic with two inputs at the bottom. If input `Sq0` is 1 then the instruction in `ID`-stage slot 0 will be squashed, likewise for `Sq1`.

In this implementation fetch groups are not aligned.



(*a*) When a branch is taken we may need to squash one or two instructions (the number of instructions to squash depends on the whether the branch is in slot 0 or slot 1). Design logic to set the `Sq0` and `Sq1` inputs so that appropriate instructions are squashed. It will be very helpful to draw pipeline execution diagrams showing a taken branch in slot 0 and slot 1.

☑ Draw PEDs for the two cases.

Solution appears below. If the branch is in Slot 0 (Case 0) then both IF-stage instructions are squashed, if the branch is in Slot 1 (Case 1) then only the slot-1 instruction in IF is squashed.

☑ Add hardware to set `SQ` signals.

The solution appears above. Notice that the squash signals are put in the pipeline latch so they move with the `IF`-stage instruction.

```
#                   SOLUTION - Case 0: Branch in slot 0.
#  Cycle            0  1  2  3  4  5  6  7
 sw r7, 4(r15)      IF ID EX ME WB
 addi r15, r15, 4   IF ID EX ME WB
 beq r1, r2,  TARG     IF ID EX ME WB
 add r3, r4, r5        IF ID EX ME WB
 slti r16, r17, 8         IFx
 or r18, r19, r20         IFx

 xor r10, r11, r12
TARG:
 sub r7, r8, r9              IF ID EX ME WB
 lw r14, 0(r15)              IF ID EX ME WB
#  Cycle            0  1  2  3  4  5  6  7


#                   SOLUTION - Case 1: Branch in slot 1.
#  Cycle            0  1  2  3  4  5  6  7
 sw r7, 4(r15)      IF ID EX ME WB
 addi r15, r15, 4      IF ID EX ME WB
 beq r1, r2,  TARG     IF ID EX ME WB
 add r3, r4, r5           IF ID EX ME WB
 slti r16, r17, 8         IFx

 xor r10, r11, r12
TARG:
 sub r7, r8, r9              IF ID EX ME WB
 lw r14, 0(r15)              IF ID EX ME WB
#  Cycle            0  1  2  3  4  5  6  7
```

(*b*) Notice that the branch hardware shown can only provide the target for a branch in slot 1. Add hardware for providing the branch target of a branch in slot 0. Note that unlike the final exam, in this problem fetches are not aligned. That precludes the more efficient solution given in the final exam.

Do not add hardware for checking the branch condition. Show logic computing the select signals for any multiplexors you add, but do not show any other control logic. *Note: In the original assignment the direction to show logic computing select signals was omitted.*

☑ Add hardware for a slot-0 branch.

☑ **Pay attention to cost**.

☑ Be sure the hardware computes the correct target address. Think about the value of `NPC` (or related value) that's needed.

Solution appears below. The slot-1 immediate is selected by the upper-left green mux if there is a branch in slot 1, otherwise the slot-0 immediate is selected (and is ignored if neither slot holds a branch). If the branch is in slot 1 then `ID.npc` is the branch PC plus 4, which is just what we need to calculate `PC+4+imm*4`. (The actual computation is effectively `4*( (PC+4)/4 + imm )`.) If the branch is in slot 0 then `ID.npc` is the branch PC plus 8. In order to compute the correct target 1 is subtracted from the immediate. That is, we compute `4*( (PC+8)/4 + imm - 1 )`.

Note that by subtracting 1 from the immediate we only need a 16-bit subtractor. Had we naivley subtracted 4 from `ID.npc` we would need a 30-bit (or even a 32-bit) subtractor.

In the final exam fetch groups were aligned and so a lower-cost solution was possible. See the final exam solution.