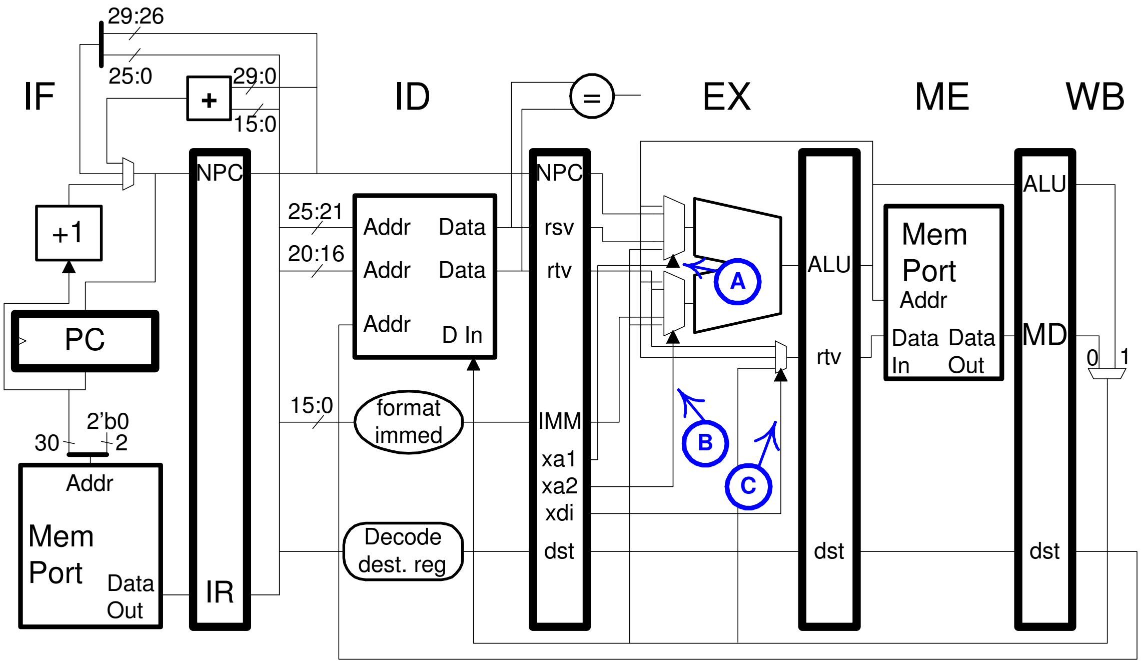
LSU EE 4720 Homework 2 Due: 1 October 2007

For lecture material relevant to this assignment see `http://www.ece.lsu.edu/ee4720/2007f/lsli06.pdf`*. For some background and a list of similar problems see the statically scheduled study guide,* `http://www.ece.lsu.edu/ee4720/guides/ssched.pdf`*. Please make an effort to solve this problem based on an understanding of the material, use the solution to similar problems (if any) only for hints. Feel free to ask questions using the forums, E-mail, or in person. Exam problems will be based on the assumption that students completed (really completed) homework assignments, so don't short-change yourself!*

Problem 1: Consider the following MIPS code and implementation:

```
 # Cycle             0  1  2
 lw r2, 0(r10)       IF ID EX
LOOP:
 lw r1, 0(r2)           IF ID
 add r3, r1, r4
 sw r3, 4(r2)
 bne r3, r5  LOOP
 addi r2, r2, 8
 # Cycle             0  1  2
A:                         2
B:                         2
C:
```



(*a*) Complete the pipeline execution diagram of the execution of the code above on the implementation illustrated for at least the first two iterations. (See the next part for instructions on the "A:", etc.)

(*b*) After the `addi` instruction three labels are shown, `A:`, `B:`, and `C:`; similar labels are shown, in blue and circled, in the implementation. On the pipeline execution diagram show the values on the wires (which are multiplexor inputs) that those labels point to *only in cycles in which those signals are used*. The values are already shown for cycles 0, 1, and 2. Signals `A` and `B` are used in cycle 2 (but not 0 or 1), signal `C` is not used in cycles 0-2.

Note that the multiplexor inputs are numbered from the top starting at zero.

(*c*) Find the CPI of this loop on the illustrated implementation for a large number of iterations.

(*d*) Add bypass connection(s) so that the loop above executes as quickly as possible. Show the CPI with those connections.

(*e*) Even with bypass connections the loop above, regrettably, executes with stalls (or at least it should!). Schedule (re-arrange) the code so that it executes without stalls. The scheduled loop should still load and store one value per iteration. Minor changes to the code can be made, such as changing register numbers and immediate values.