Assignments 5, 6, 7

Task: Incorporate a translation lookaside buffer in the `mips` design. There will be two TLBs, one for instructions, `iTLB`, and one for data, `dTLB`. This allows them to be smaller and hopefully faster. [This assignment is somewhat underspecified on purpose.]

Requirements: The present `mips` design has the following template:

`|-if-|-id-|-ex-|-mm-|-wb-|`

and we should expect that adding the address translation which must come ahead of the cache access to add a stage for each memory reference. However, we can with careful revision of the design take the extra cycle only at the data fetch, yielding the template

`|-if-|-id-|-ex-|-m1-|-m2-|-wb-|`

1) A `userid` field must be added to the design. For design purposes it can be set to one initial value.
2) Revise your TLB design so that the output is timed to take 5 ticks. (See the `imem` and `dmem` modules for how this might be done.)
3) Initialize your TLBs to contain a few of values. (We will not implement the instructions that set the TLB, so we assume that they always have the values in them that they need.) Be sure these have in the userid field the value you assigned to your test userid.

iTLB

4) [Refer to the pipeline schematic in H&P Chapter 6.] Replace the cpa module in the `ifetch` stage with a special module `cpa4` that adds 4 to its argument and does so in 5 clock ticks.
5) In the `ifetch` stage, insert an `iTLB` lookup after the `pc+4` operation. The physical address returned will be used to reference the `imem`. Remember the virtual address still has to be passed along.
6) The branch address calculation that previously occurred in the `ex` stage is moved ealier to the `id` stage.
7) The `iTLB` look up for the branch is performed in the `ex` stage. Both the virtual and the physical addresses are passed to the memory stage. Note that this `iTLB` look up takes place in the first part of the stage, while the `iTLB` lookup of (5) is to be performed in the last part of the stage.
8) Clean up any vestigial parts of the simulation that are no longer needed. Test this `mips` design with instruction TLB. It should work pretty much the way the old one did, except that the addresses are physical.

`dTLB`

9) Split the memory stage into two parts, `m1` and `m2`. The requires both that you develop the proper modules and the logic to pass the pipeline registers along. The memory operations that were previously performed in the mem stage move to `m2`. Branching related operations are to be in `m1`.
10) The `dTLB` lookup is to occur in `m1`, with the physical address transferred to `m2` to be used in the memory operations.
11) Update the instruction stalling/flushing logic to accomodate one additional stage of the memory delay.
12) Clean up any vestigial parts of the simulation that are no longer needed. Test the mips design with data TLB.
13) The extension of the pipeline has increased the interlock distance for data hazards by one pipeline stage. What will be the impact be on instruction execution.

Task: Incorporate forwarding to recover the cost of stalling for data hazards. The complete discussion of forwarding for the base `mips` design is discussed in Section 6.6 in H&P. See especially Figure 6.43.

Forwarding in the base design.
14) Following the H&P discussion, incorporate forwarding into the `ex` stage of the basis `mips` design. ALU operands can possibly come from the `mem` stage or the `wb` stage. The logic of the ALUSelA and ALUSelB conditions must be computed and the proper operands muxed into the `alu`.
15) Clean up the design of any vestigial code. Test the new design showing that it works.

Forwarding in the TLB design.
16) Generalize the ALU operand muxes and the ALUSel control signals to handle the added memory stage introduced for dTLB. How may possible input sources are there? What are the logic equations for determining where the source of the operand comes from.
17) Incorporate the forwarding generalization into the TLB design. Note, it will still be necessary to stall instructions following some loads since the data value simply is not available.
18) Clean up the design of any vestigial commands. Test the new design showing that it works.

Task: [Optional Extra Credit] Incorporate the BTB into the `mips` pipeline. The lookup takes place in the `ifetch` stage after the `pc+4` is computed, causing it occur in the last part of the stage. Supply all further details including use of the prediction, verification and pipeline flush.

Turn in: Since the basis design will largely remain unchanged, each part of this assignment is to be answered by giving portions of the base design that were changed together with an explanation of what the change is intended to accomplish. Think of the answers as your log of the changes. <u>All</u> changes are to be listed. In addition, a completed design for each task (4 versions of `pipeline.v`) is expected.

Due date: The entire assignment is to be turned in by 5:00 pm on 9 December 1996. Prior to that, either 6 parts are to be turned in by 5:00 26 November 1996 OR 9 parts by 5:00pm 2 December 1996 [These are graded, credit/no credit . The actual solution can be improved for the final submissiion.]