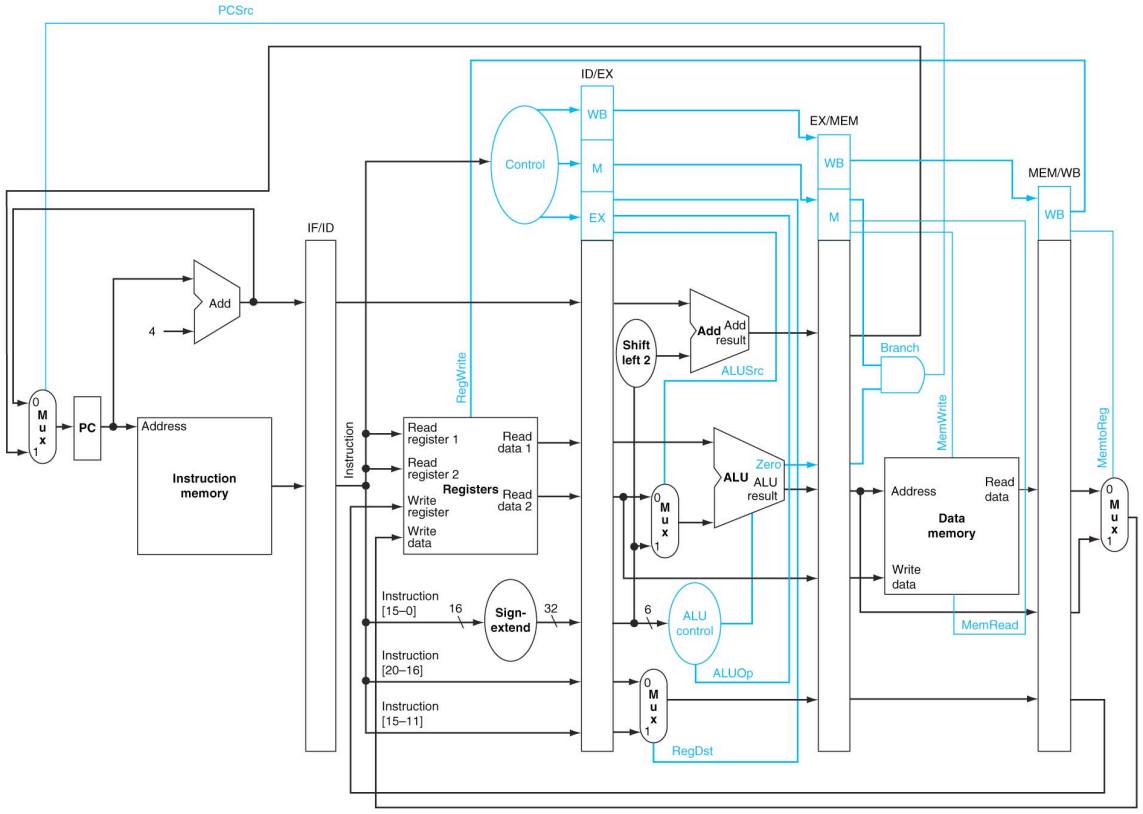
CprE 381 – Computer Organization and Assembly Level Programming
Fall 2015

Mini-Project B

*Summary: This is a **three-week project**. You will design and implement a pipelined MIPS CPU based on your single-cycle MIPS CPU from Mini-Project A. The project involves substantial design, implementation, integration, and test tasks.*



Textbook Figure 4.51

Part1: The Simple CPU pipeline

Overview You will implement the simple pipeline similar to Figure 4.51 in the textbook. Start from you single-cycle processor, insert pipeline registers to make it a simple pipeline (without any hazard handling).

Note: Your SCP is more complex than the textbook one, so your pipeline will be more complex than what the diagram shows.

Starting Point You are provided with the following VHDL files in the attached .zip file. Add those files to your working directory of Mini-Project A, and then add them to your ModelSim project.

1. **cpu_pl.vhd**: Template file for modeling the organization of the pipelined CPU. You may revise it as you wish, but you are suggested not to change the structure of the file.
   Note: There will be an update on this file to support better debugging.
2. **pl_reg.vhd:** Template file for modeling of the pipeline registers. You may add fields to each pipeline register. Note that it uses VHDL record type to aggregate the fields of each pipeline register.
   Note: There may be an update on this file to support better debugging.
3. **tb_cpu3.vhd** (*to be included*)**:** The test bench program that uses your cpu0, the behavioral modeling of MIPS CPU, to verify your pipelined implementation. *This file will be released no more than one week later.*
4. **cpu_entity.vhd:** This is for an updated CPU entity, for checking the pipelined CPU with the reference CPU. Use this file to replace the existing cpu_entity.vhd. Note: It works with your current *behavioral* and *scp* architectures.

Additionally, you are suggested to add your load-use hazard unit, data forwarding unit, and a branch resolve unit to your current **small_elements.vhd**. The branch resolve unit detects any taken branch or jump and triggers pipeline flushing.

VHDL Modeling Requirements

The VHDL coding for the above entities **must be strongly structural** as in Project B. Specifically:

1. **No behavior modeling** for the CPU organization. In particular, no process statement can be used.
2. **Limited dataflow modeling**. The following three forms are accepted:
   - Signal copying/splitting, e.g.
     ```
     ID_opcode <= inst(31 downto 26);
     ```
   - Signal merging, e.g.
     ```
     EX_j_target <= PC(31 downto 28) & j_offset & "00";
     ```
3. All entities being used must be declared as component. Only *component instantiation* can be used; no entity instantiation may be used.

Additionally, whenever appropriate, you must prefix signal names with a name to indicate pipeline stages. For example, use EX_j_target for a jump target signal at the EX stage.

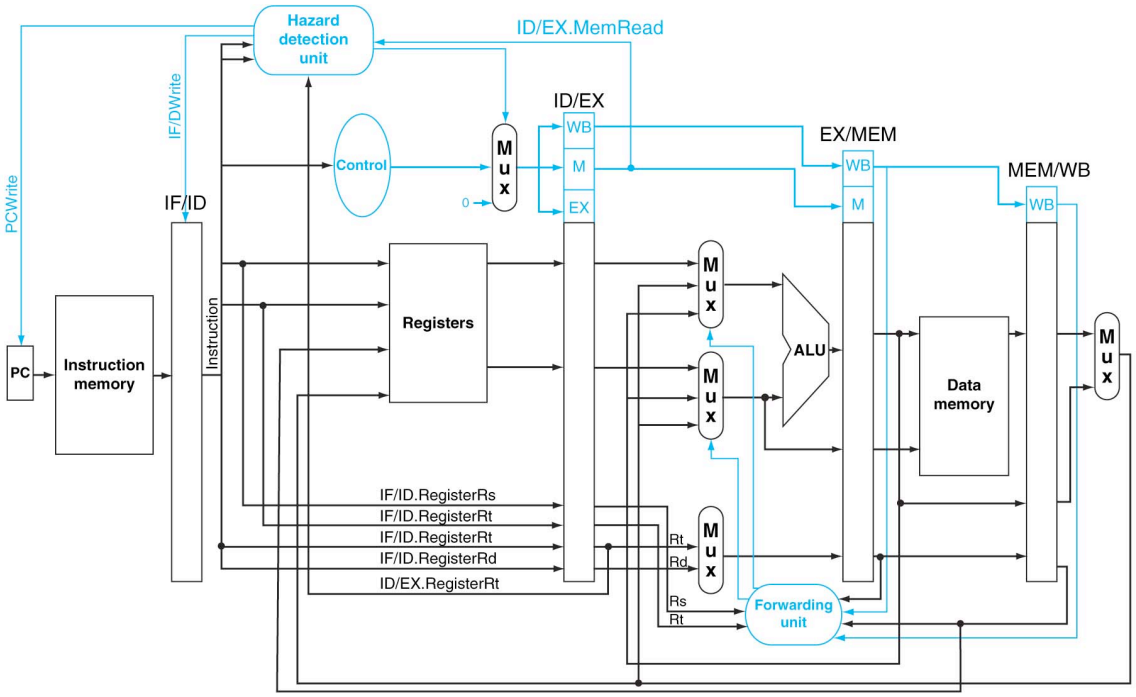
Demonstration Create one or more testing codes for your simple CPU pipeline. *The MIPS testing code should not carry any potential pipeline hazards to the MIPS pipeline, or you have to insert nops*. The following is an example of such MIPS testing code:

```
addi  $a0, $zero, 0x0010
addi  $a1, $zero, 0x0020
addi  $a2, $zero, 0x0004
addi  $a3, $zero, 0x0008
sll   $t0, $a0, 2
add   $t1, $a0, $a1
lw    $t2, 8($a2)
sw    $t3, 4($a3)
```

Part 2: CPU pipeline with data forwarding and hazard detection.

Implement the data forwarding unit and a data hazard detection unit, as demonstrated in the following diagram. Note that your CPU is more complex than what the diagram shows.

Note: The template file cpu_pl.vhd contains those components.



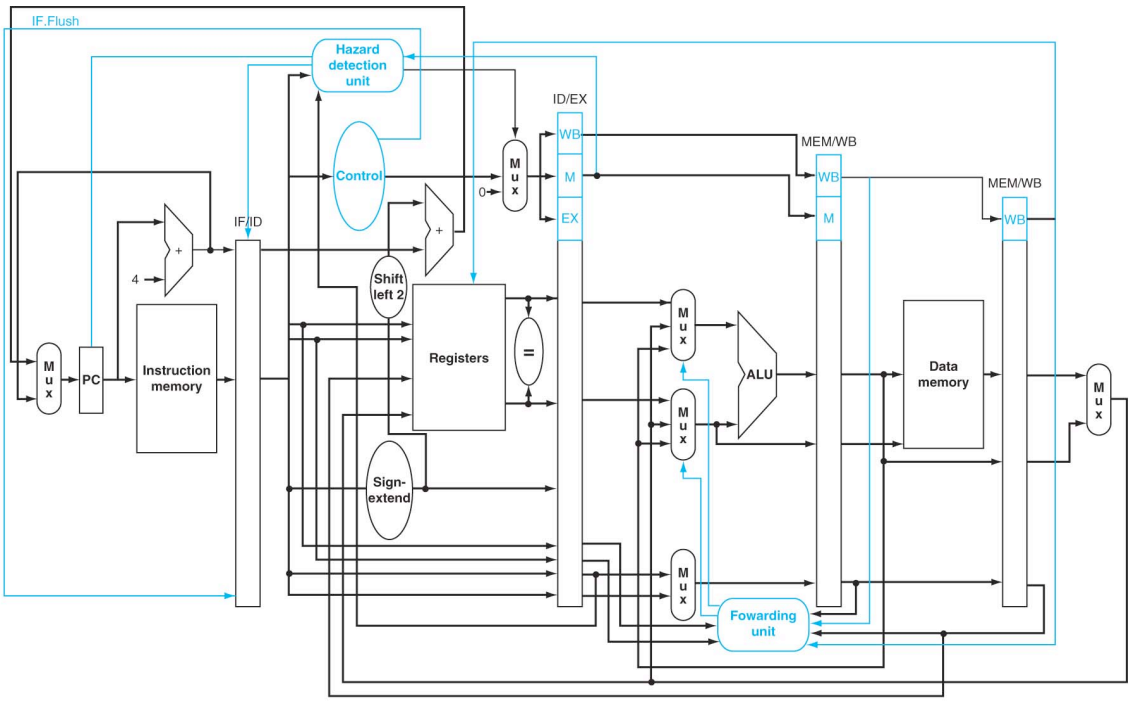
Textbook Figure 4.60

Demonstration Create testing codes that contains ALU and load-use hazard. Demonstrate your pipelined CPU works correctly. The following is an example of such MIPS testing code (you may want to add more instructions to initialize the registers):

```
sll $t1, $a1, 2
add $t1, $a0, $t1
lw $t0, 0($t1)
lw $t2, 4($t1)
sw $t2, 0($t1)
sw $t0, 4($t1)
```

Part 3: CPU pipeline with data forwarding, data hazard detection, and control hazard handling.

In this part, you will add control hazard handling to your CPU pipeline. The following diagram shows the MIPS pipeline with branch comparison at the ID stage. Again note that at this time, your pipeline implementation is more complex than the diagram except on the control hazard handling.



P&H Figure 4.65.

Your implementation will be different. Instead of resolving branch/jump in the ID stage, resolve them at the EX stage. Then, you don't have to change the data forwarding. Note that the two instructions at the IF and ID stages will be flushed for every taken branch or jump.

Demonstration. Demo that your CPU pipeline can run the bubble sort code correctly. You should show the progress that an array in the data memory gets sorted. In the demo, you may also be asked to show certain signal values for more correctness check, and you may be asked to explain your design.

DEMO AND SUBMISSION:

- Print out a hard copy of the evaluation from (Project-B-*Eval.pdf* on the class web site).
- If you can demo part 3, you do not have to demo part 1 or part 2.
- Complete the demos to your TA and ask your TA to sign on the evaluation form.
- Create a zip file *Project-B-submit.zip*, including the completed code in your project (all files from Mini-Projects A and B).
- The file names in your zip file should be self-explained.
- Submit the zip file on BlackBoard Learn under "Project B" assignment.