ECE/CS 552: INTRODUCTION TO COMPUTER ARCHITECTURE

Project Description – Stage 2

First Stage Project Report Due: ~~**November 24, 2010**~~
Final Stage Project Report Due: **December 15, 2010**
Project Demonstrations: **December 11-12, 2010**

The project should be completed in a group of three students.

1. Review of the First Stage Project

Before you start reading the second stage project description, you should have a working design in the first stage that has the following 16 instructions implemented:

Table 1: Table of opcodes

Function	Opcode
ADD	0000
SUB	0001
AND	0010
OR	0011
SLL	0100
SRL	0101
SRA	0110
RL	0111
LW	1000
SW	1001
LHB	1010
LLB	1011
B	1100
JAL	1101
JR	1110
EXEC	1111

To see the detailed information of the ISA, please refer to the first stage project description. You cannot proceed if you do not have a working design for these 16 instructions.

2. Memory System

In the second stage, the instructions and data will share a main memory, which has a latency of 3 cycles for read and write. You should also implement an instruction cache and a data cache with one-cycle delay. Both caches are direct-mapped, and have a block size of 8 words (16 Bytes) and a data array size of 512 words (1024B organized as 64 lines). Therefore, The address[2:0] is the block offset, address[8:3] is the index and the remaining bits in the address are tag bits. Besides the tag bits, the tag array of your cache should have one valid bit per entry indicating if the cache line is empty (0 means invalid, 1 means valid).

Figure 1 shows the overall memory system diagram of WISC-F10. In this figure, control logics and signals are not shown.

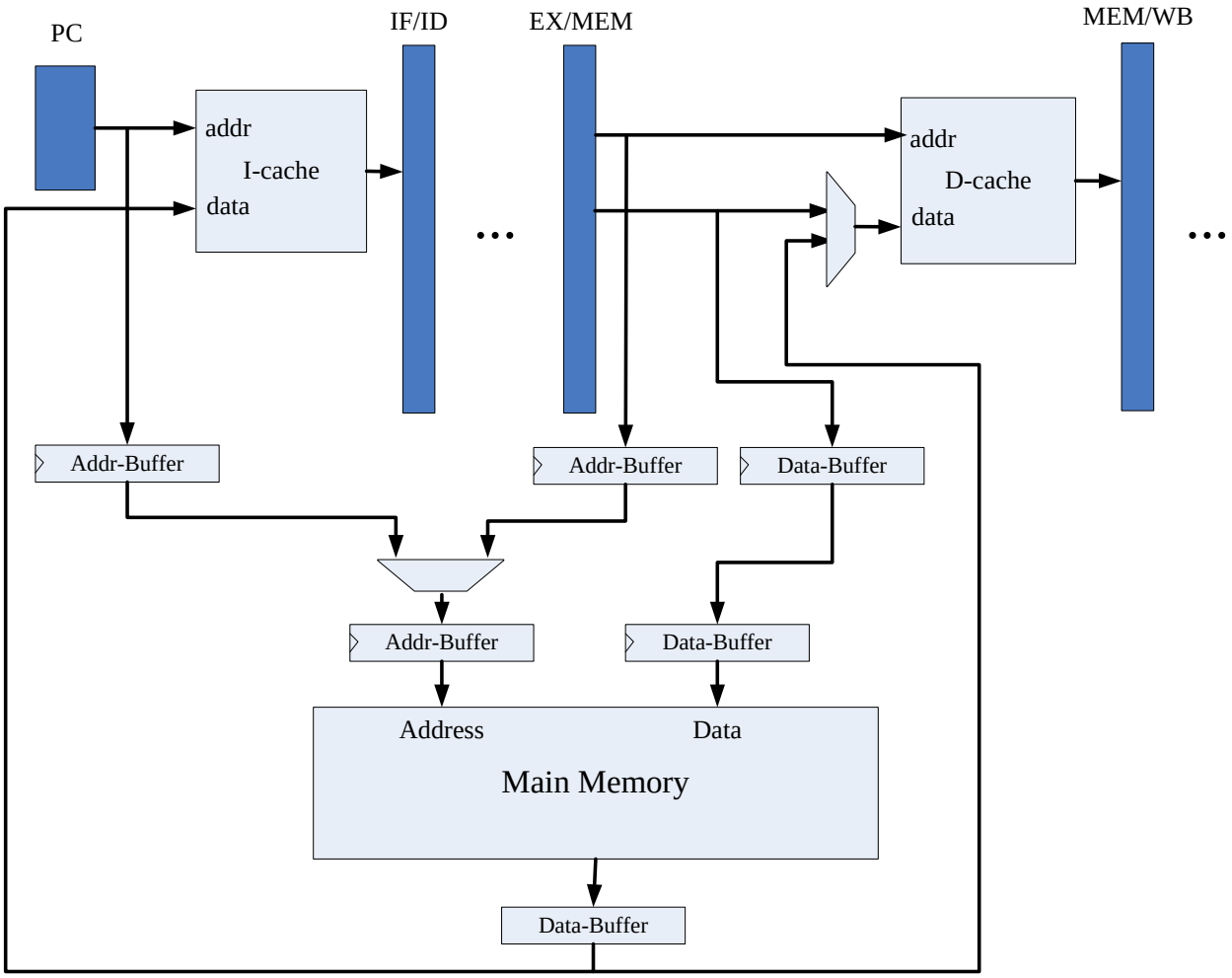

Figure 1: Memory system

From figure 1 we can see that I-cache and D-cache share one main memory. For the instruction cache, when there is a fetch hit in the I-cache, the pipeline will not be stalled. When there is a fetch miss in the I-cache, the address is stored in the address buffer of the main memory after 2 cycles. The address buffer should hold the address for three cycles until the main memory fetches the instructions out and store them in its data-buffer. After that, the data buffer will write the instructions to the I-cache. You should generate the write-enable signal for the I-cache and update the tag array appropriately.

Since your I-cache has an 8-word line, you must fill the entire line from the memory to the cache. Therefore, the read-miss delay is 8 times of the delay for reading one word.

You can assume that the I-cache is never written by the processor. Writes to the I-cache only happen on fetch misses.

For the data cache, the read operation is similar to the I-cache. For writes, you will implement a write-through no-allocate policy. When there is a write instruction (SW) in your pipeline, your cache needs to check if it hits in the cache. If it is a write hit, the data-cache will write the word to the cache and also to the main memory. If it is a write miss, the processor also writes to the cache and memory, but mark the cache line as "invalid" (since the data was written to a line containing data from a different address).

Since you only have one 16-bit write bus to the memory and each write takes multiple cycles, you cannot execute "SW" instructions back-to-back. Therefore, you need to have a hazard detection unit that separates SWs by inserting NOPs in your pipeline. This is a structural hazard detection unit. Figure 2 shows the correct operation when two SW instructions are back-to-back.

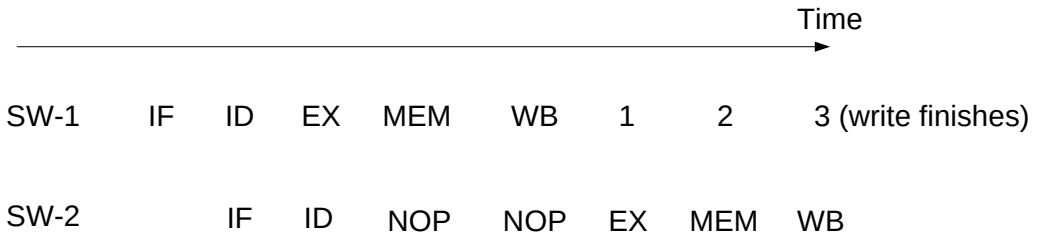

Figure 2: Hazard detection for SWs

Since I-cache and D-cache share main memory, they cannot access the memory concurrently. You need to implement selection logic to pick one that accesses the memory. You can either give priority to one cache over another, or implement a locking scheme that guarantees mutual exclusion.

Verilog modules will be provided for all the memories and caches. You can use a 1-cycle module to instantiate tag array and data array for your caches, and 3-cycle module to instantiate the main memory.

You will get bonus credits if you can implement extra features that enhance the performance of the cache. You may not change the total size of the data array of the caches, or the width of the bus (16 bytes) between the L1 caches and main memory. You may implement set-associative caches, a write-back policy, fully-pipelined stores, memory interleaving, critical-word-first mechanism and so forth. Discuss your ideas with the TA or instructor before proceeding.

3. Extra features

You will get bonus points if you implements extra features to enhance the power of WISC-F10. As a general guideline, a very simple additional feature might add 1% to your project grade, while it is unlikely that any combination of advanced features will cumulatively exceed a 25% increase to your project grade. Since the project is worth 20%, this means your final grade will improve by 0.2% to 5%, which may then boost you over the threshold to the next higher letter grade (it is unlikely it will boost you by two letter grades). The extra features must satisfy the following requirements:

1) You must prove that your extra feature does enhance the power of the computer. Do not implement extra features that degrade the performance or computational capability of your design.
2) The extra feature must not break the baseline. Otherwise you must have a copy of the baseline implementation and show both in the demo.
3) It must be somehow non-trivial. The effort that you spend on this extra feature determines the points you may receive.
Most importantly, **Make sure you discuss any extra features with the TA and instructor before proceeding.**

4. Submission Requirements

Second Stage Project Report

The project should be done in a group of three. The second stage project report should emphasize the new elements that added to your design since the first stage of the project. The second stage project report should include the following parts:

1) A brief description of or an introduction to your project. Include a top-level schematic (if you write Verilog at top-level, you may draw a block diagram of you top-level design. Hand writing is okay.) Report its special features, any particular effort you have made to optimize the design, and any major problems you encountered in implementing the design.
2) A statement indicating whether or not your design meets all the requirements specified in this document.
3) Include schematic printouts or Verilog listings of major blocks that are not included in first stage project report.
4) The simulation results for test programs (will be provided later) in "list" form. Be sure to include the following signals:

   a. Instructions fetched from I-cache/memory.

   b. ALU operand, ALU result and ALU control signal (opcode).

   c. Register file read and write address and data; register control signal RegWrite.

   d. Memory read and write address and data; Memory control signal MemWrite (or WE).

   e. Data being written back at WB stage.

   These signals may come from the pipeline registers. Highlight the above signals, and add more in your simulation result if necessary.
5) A detailed description of the methodology you used to test the correctness of your design. Include a well commented copy of the key programs you used to create test inputs and/or verify the outputs from the various blocks.
6) If your project implements extra features (forwarding, etc.) that are not included in first stage project report, provide a short paragraph highlighting the extra features. Also, provide an estimate of performance enhancement due to these extra features for the testing program. Bonus point will not be considered if the design is incorrect.

Project Demonstration

Each group will be asked to demonstrate their project on **Saturday or Sunday December 11-12, 2010**. All team members must be present at the demo and must be prepared to answer detailed questions about the design. During the demonstration, you will:

1) Show the overall design of your project.
2) Run a few test programs, including the ones provided before hand, and the ones that will be provided during the day of the demonstration. To get full points, your design must give correct results of both types of programs.
3) Answer the questions that instructors may have based on the simulation result.
4) Demonstrate the extra features you may have in your design.