ECE 445
Computer Organization and Design
Spring 2009

MP2: The LC-3b Processor

1 Introduction

Congratulations on completing MP1. You now have the design of a preliminary LC3 processor. However, it only implements 6 of the LC-3b instructions. In MP 2, you're going finish what you started in MP 1, and implement the rest of the LC-3b instructions. When you are done with MP 2, you'll have a fully working LC-3b processor!

Unlike the previous MP, this handout is only intended to help you get started. You are responsible for implementing the additions and modifications to the MP1 design on your own. We will cover datapath, control, and cache design in lecture. In addition, Section 5.4 of Patterson and Hennessey provides an example of another design, which might supply you with ideas on how to proceed.

At the end of this MP, you will deliver a working LC-3b processor. Grades will be based on the correctness of design, style of design, thoroughness of verification, and quality of documentation.

2 Before You Start

This section is meant to prevent and alleviate problems you don't want to encounter as the MP progresses.

2.1 Blackboard Discussions

The course's Blackboard discussions are the best resources you have to get help and tips. The TAs monitor this resource regularly (at least twice a week), and often fellow students will answer your question. Be sure that you have read all previous posts before posting your question; your question may have been addressed already.

You may save yourself a lot of time and trouble by regularly checking the Blackboard discussions. By learning from other student's problems, you can save yourself from making the same mistakes. Also, tips, updates, and clarifications from TAs will help make your MP experience more pleasant. You are responsible for any updates or announcements posted on the Blackboard, so make sure you check the course's Blackboard page regularly!

3 The LC-3b Instruction Set Architecture

For this project, you will implement the rest of the LC-3b instruction set architecture (except the RTI instruction). For a complete description of the LC-3b ISA, refer to the ISA reference manual available on the course's Blackboard page.

All sixteen instructions are 16 bits in length, having a format where bits [15:12] contain the opcode. The LC-3b ISA is a load-store ISA, meaning data values must be brought into the General-Purpose Register file before they can be operated upon. Each general-purpose register is 16 bits in length (meaning that the LC-3b is a 16-bit ISA). The address space of LC-3b is accessed using 16 bits (meaning that the LC-3b memory consists of 2^{16} locations). Each location contains a single byte (meaning that the LC-3b memory is byte-addressable). The General-Purpose Register file contains 8 registers.

As in MP1, the LC-3b program control is maintained by the Program Counter (PC). The PC is a 16-bit register that contains the address of the next instruction to be fetched.

4 The LC-3b Memory Control

4.1 Memory Control Signals

Your microprocessor design will use the same signals to communicate with the outside world as the processor of MP1 did, with the exception of the MWRITE L signal. For this and future MP's, the MWRITE L signal of MP1 will be split into two signals, MWRITEH L and MWRITEL L. Descriptions of these signals are provided below.

MWRITEL L Active low signal that tells memory that the address is valid and that the processor is trying to perform a memory write to the lower byte of a word.

MWRITEH L Active low signal that tells memory that the address is valid and that the processor is trying to perform a memory write to the upper byte of a word.

4.2 Memory Control Logic

The Bus control logic is similar to that of MP1, again with the exception of the MWRITE L signal. The processor sets the MWRITEL L and/or the MWRITEH L signal(s) active when it is writing to the memory. We still assume the memory response will always occur so the processor never has an infinite wait period.

5 Getting Started

You will be using your MP1 design as the base for MP2. Since you are working in a group of two students you should choose the cleanest implementation to start your MP2. You should first copy your MP1 design to make a backup just in case. To archive your design, select **Design → Archive**, then select all your source files.

If you look in the vhd file for the LC3b types (you can access it through the design browser), you will see that it defines subtypes and constants. Subtypes are like C++ typedefs, and constants are like C constants. Notice that types for imm5, trapvect8, etc. are already provided for you. So when you are creating a bus (wire), give it one of these provided types instead of specifying std_logic_vector().

You may need to create your own types. For instance, for a mux select that requires a 2 or 3-bit select input, you may wish to define a type for the input. You will need to add constants for the rest of the opcodes. You will also need to add constants for additional ALU opcodes. And finally, you will need to add some more constants for time delays.

Finally, simulate the entire design that you copied over to ensure that there are no errors before continuing. This is VERY IMPORTANT.

6 Design Limitations ***IMPORTANT***

6.1 Things You Must Not Do:

- Begin MP2 until you are absolutely convinced your MP1 works.
- Add any additional ports to the LC-3b register file.
- Add additional datapath registers (such as IR, PC, MDR, MAR).
- Add additional inputs to the ALU.
- Design non-realistic or behavioral VHDL. For example, muxes should be nothing more than a case statement.
- Use the same output from the IR for all the immediate values. Imm5, TrapVect8, Index6, Offset9, and more are provided for you in the types package.
- Add logic in-between the MDR and DATAOUT bus (except multiplexers).
- Add logic in between the DATAIN bus and MDR (except multiplexers). This also means you may not add inputs to the MDRMux.

6.2 Things You May Do

• Make additional muxes with the following limitations:

i. A MUX must have one data output. This output must be the same width as the inputs.

ii. A MUX select signal must have 1, 2, or 3 bits.

iii. Appropriate delays (based on number of data inputs) must be used.

a. The delay for a 4 input MUX is 2 times that of a 2-input MUX.

b. The delay for an 8 input MUX (including 5, 6, and 7 input variations) is 3 times that of a 2-input MUX.

iv. The value that appears on the data output must be a value that appears on one of the data inputs. No shifting, sign extension, zero extension, or other magic inside a MUX. Exception: One or more hard-coded value(s) may be declared inside a MUX. Each hard-coded value counts as a data-input.

• Add logic gates with the following limitations:

i. Functions allowed: AND, OR, NOT, NAND, NOR, XNOR, XOR.

ii. Limited to a maximum of 4 inputs per gate. (2/gate for XNOR and XOR.)

iii. All inputs to a given gate must be the same width.

iv. Exactly one output. The output must be the same width as the inputs.

v. Delay is based on the number of inputs.

a. The delay for a 2-input gate is 1 ns.

b. The delay for 3 and 4 input gates is 2 ns.

• Add comparators with the following limitations:

i. This component has two data inputs. Each input must be the same width.

ii. Exactly one data output. It must be 1 bit. This output signals equality of the inputs. You may choose active high/active low for the output.

iii. You may choose to use a hard-coded value to replace one of the data inputs if you wish.

iv. Comparator delay is independent of the width of its inputs:

a. The delay for a comparator is 5ns.

• Restrictions on state-machine control units:

i. Your state machine must be synchronous.

ii. No actions on transitions.

iii. Your machine should be a Moore model machine. That is, outputs should only depend on the current state, not the current inputs.

iv. Complex conditions on transitions are allowed. That is, a condition may contain several AND's or OR's, or combinations of them.

• Add additional ports between the Datapath and Control components.

• Lengthen your clock to as much as 50 ns to accommodate your critical path. Make sure that the MRESP H signal in Memory.vhdl is held high for the length of your clock cycle. Make sure the START H signal in your test bench is held high for the rising edge of one clock cycle.

• Sign-extension has the delay of a two-input mux (this MUX should use the most significant bit to select between 0's and 1's).

• Zero-extension and other Concatenations and Splittings all require no delay.

i. These can all be thought of as ways to route wires. Concatenation, for example, is taking some wires from one bus and some wires from another bus and forming a new bus from them.

ii. Boxes that perform these functions should perform no logic.

For example, no boxes that sign-extend or zero-extend, depending on some select signal. If you want to do this, make a box that sign extends, a box that zero-extends, and select between their outputs with a MUX.

6.3 Things You Must Do

- Sign-extension must be represented by its own block and VHDL code.
- Delays need to be coded into your individual component files. Hence, symbol provided by ActiveHDL must not be used because it does not have its delay coded).
- Make sure the MWRITE and MREAD signals are not getting set low (turned on) until after the address (and data) have stabilized. This may be accomplished by increasing their delay if necessary.
- Ensure that you are never trying to read from and write to memory at the same time. That is, MWRITE and MREAD must not ever be active at the same time.
- Write your own programs in LC-3b assembly to thoroughly test and debug your design.

7 Hand-in Requirements

Exact hand-in requirements will be posted on the course Blackboard page (at least) a few days before the due date. Part of hand-in will require you to run our test code. We will be making this test code available on the Blackboard page shortly before the due date. You should not delay debugging your design until our code is released; write your own code to test your design's correctness.