CprE 381 – Computer Organization and Assembly Level Programming

Project A

In this two-week, group lab assignment you will implement a single-cycle MIPS processor. The architecture shall support at least 6 MIPS standard instructions: ADD, ADDI, LW, SW, BEQ, J. This is the minimum set of instructions to support the full control/data path presented in lecture. All components for the processor are provided with the exception that you will have to update the main control unit as part of the project.

0) Before working on the lab:

(a) Create a new folder for project A <user home folder>/cpre381/projectA. Use this directory to save your work.

(b) Unzip the provided projectA.zip in the project A folder. The zip contains:

a. "Components":

- adder_32.vhd
  - A simple 32-bit adder that takes two input and outputs one result. This could be useful for calculating PC + 4 and/or PC + branch target offset.
- ALU.vhd
  - ALU that supports 12 operations. This is the same ALU from lab 7.
- and_2.vhd
  - A simple 1-bit, two-input AND gate. Useful for ANDing the branch control flag with the ALU zero flag to resolve branches.
- dmem.vhd
  - A data memory component. The same as the mem component in lab 7.
- imem.vhd
  - A instruction memory component. The same as the mem component in lab 7.
- main_control.vhd
  - Main control unit for parsing an instruction into control flags used in throughout processor. Contains an implementation for the ADD instruction.
- mux21.vhd
  - A 32-bit, 2:1 mutliplexor.
- pc_reg.vhd
  - A simple 32-bit register which can be used as the program counter.
- register_file.vhd
  - 32, 32-bit registers. The same interface as the register file from lab 7.
- sign_extender_16_32.vhd
  - 16-to-32 sign extender for I-type instructions with an immediate.
- sll_2.vhd

- A shift-left-logical unit that always shifts the input 2 places.

b. dmem.mif (a data memory initialization file)
- Fills data memory with dummy values on initialization

c. imem.mif (an instruction memory initialization file)
- Contains a simple MIPS program for testing your processor

Get a good understanding of all the components provided so you can use them to complete the project.

1) A **Single-Cycle MIPS Processor** is a simple RISC architecture that executes any arbitrary standard MIPS instruction in one clock cycle. For this project you will implement a single-cycle MIPS processor. You may combine the provided components (register file, ALU, memories, …) to build the architecture using Quartus or structural VHDL. For design reference, see the diagram below from the textbook of the single-cycle MIPS architecture.

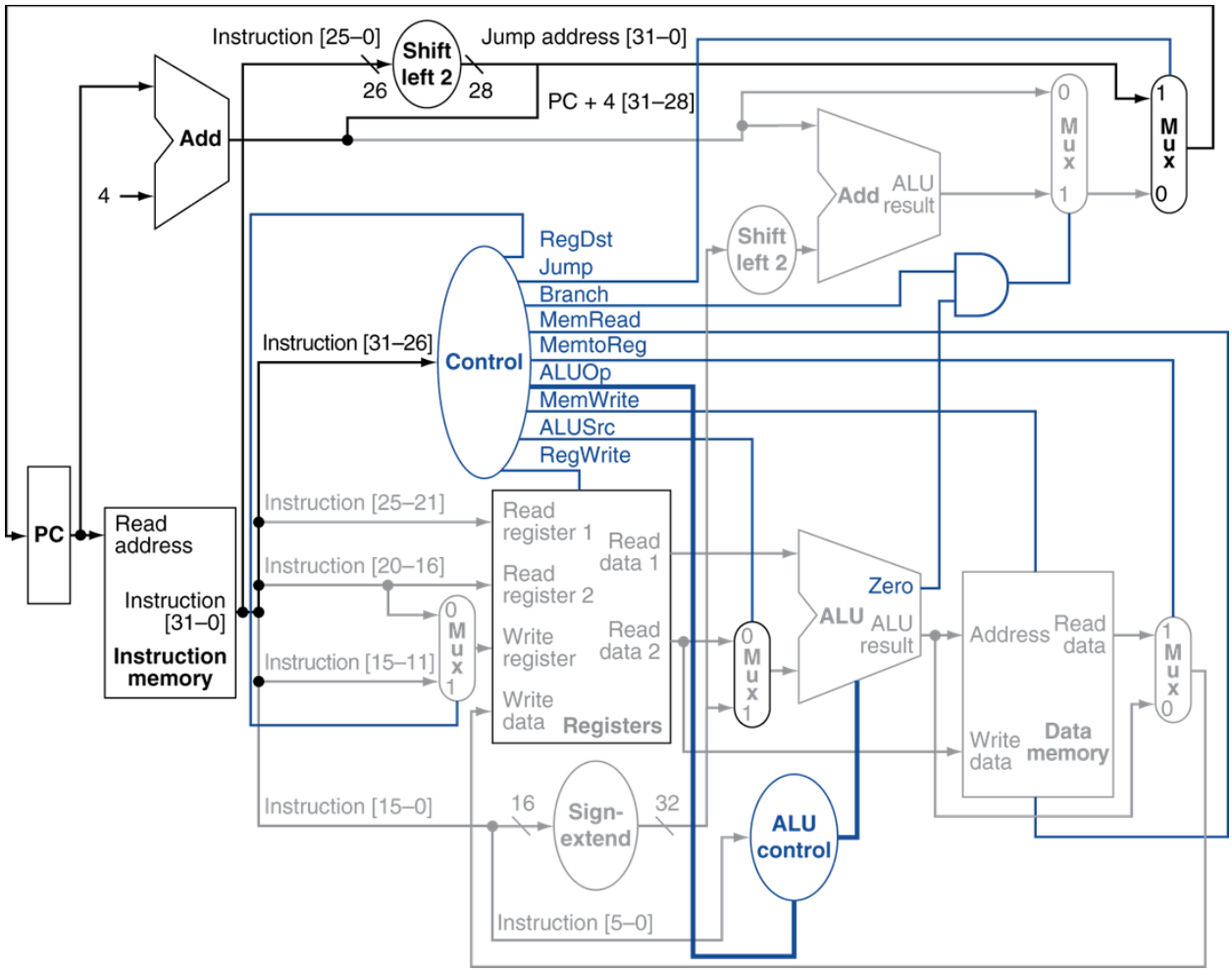

Figure 1: Single-Cycle implementation of the MIPS architecture.

A few notes before you get started:

- The memory units are word-addressable, whereas MIPS is typically byte-addressable memory. To work around this use bits [11..2] of the ALU output as the address to the data memory. For the instruction memory use bits [11..2] of the PC register output as the address. As in lab 7, set the *byteena* port of both memories to all 1s. This forces writing of entire words instead of specific bytes.

- The two memory units (instruction and data) contain the same implementation. They've been renamed to imem and dmem so that you can use different memory initialization files (.mif) which are imem.mif and dmem.mif, respectively.

- The ALU control unit is implemented internally within the ALU component. The ALU takes the ALUOp as input as well as the shift amount field from the instruction (i.e. bits [10..6] of the instruction).

- The main control unit takes the entire instruction as input. Within the main control unit the instruction is split into the op code and function code fields. This is not reflected in the diagram above.

a) Provide a description of how each of the control values in the architecture (i.e. the outputs of the main control unit) is used. How is the zero flag from the ALU used?

b) This project requires that you build the single-cycle processor to support 6 instructions: ADD, ADDI, LW, SW, BEQ, and J. The diagram provided above contains all data/control paths you need to support those instructions. Implement the single-cycle processor using Quartus or structural VHDL. You may use a combination of both if you specify your methodology/design in your report.

2) In order to verify your processor works as designed you will need to run simulation of the circuit. A memory initialization file that contains a simple MIPS program has been provided for you to test with. Simulate your processor in ModelSim and run the provided program contained in imem.mif. Provide a screenshot of the working instruction and a description of how the instruction is working in accordance with the screenshot. Do this for each of the 6 required instructions. The original assembly code can be found under ASM files → test.asm.

SUBMISSION:

- Report the names of your group members in your report.
- Create a zip file *Project-A-submit.zip*, including the completed code and screenshots from the lab.
- A lab report that answers all highlighted sections from this document. You can include your screenshots in the report if you'd like.
- The file names in your submission should be self-explanatory.

- Submit the zip on BlackBoard Learn under Project-A.