EE 6313 Advanced Microprocessors
Fall 2020 Project 1

1 Overview

The goal of this project is to design a 32-bit RISC microprocessor with load-store architecture with a 4-stage pipeline and Harvard architecture. The project also includes the instruction and data memory interfaces, register interface, and the entire pipeline control logic including full resolution of all structural, control, and data hazards.

The memory interface will only support a single asynchronous memory interface.

2 Key Requirements

Here is a list of some key features based on the architecture developed in class.

- Harvard architecture internal to the CPU, constrained to one memory bus in the memory interface until cache is added
- 4-stage pipeline (IF, RR/ADDGEN, FO/EX, WB) as shown in class
- All instructions are 32-bits long as shown in class
- ALU operations involve only register values and short immediate values stored in the instruction
- The load/store instructions shall implement all modes shown in class
- Thirty-two 32-bit registers will be supported
- Stall-based structural hazard resolution for IF, FO, and WB memory access
- Data forwarding-based and stall-based hazard resolution for register operations
- Deferred flag resolution in WB for control hazard resolution (BRA*cond*)
- The memory space in provided with a single interface with A31..A2 + ~BE3..0 addressing
- You may assume that no data access crosses an aligned quad byte boundary
- Bank enable, bus control signals, and ready need to the implemented for memory interfaces
- Assume that all instructions are aligned to a quad byte boundary
- A register structure supporting multiple-simultaneous read/write operations needs to be implemented
- All operations in the class spreadsheet shall be supported
- No ALU internal design is required this semester
- Support for an external interrupt shall be provided with the vector read on the LSB of the data bus.
- The endian-ness of the processor must be chosen and clearly stated
- The SP pointer convention (post-/pre- inc/dec on push) must be chosen and clearly stated

3 Required Components of the Design

While this list is not comprehensive, this is a list of the major components of the project submission:

a. IF stage
   This stage reads the instruction word and generates many control signals based on op code, operand modes, and data width. Stall can be caused by memory wait or by later stages in the pipeline.

   This block also maintains the tightly coupled PCfetch value, which is set to the reset vector address on reset, set to PC after PC discontinuities (such as GOTO, CALL, INT, RETI, BRA, BRAcond taken), and incremented by 4 in other cases.

b. RR/ADDGEN stage
   This stage reads multiple registers and provides these values for memory address generation or ALU arguments depending on the mode. Additionally the addresses for FO and WB are generated. This stage also selects the correct arguments for the ALU and formats (shifts) the data so that the data is LSB aligned for byte width operations or LSW aligned for word width operations. A stall can be caused by data dependency or by later stages in the pipeline.

c. EX / FO stage
   In LD and POP instructions, this stage is used for reading the data memory (fetch operand). In most operations, this stage is used for ALU calculations (execution). Stall can be caused by memory wait or memory writes in WB (structural hazard) or by later stages in the pipeline.

d. WB stage
   In this stage, writes to data registers and memory can occur. Writes to PC (except for "bubbles inserted into the pipeline) and flags are also performed for each instruction. Also, any writes to SP (for PUSH/POP) are also performed. Stall can be caused by memory wait.

e. Stall control / flush logic
   Logic is needed to detect a hazard, process memory wait states, and create the stall signals for each stage. When a stage is stalled, the input to the synchronous registers between stages must propagate an operation that does not cause any register or memory reads or writes. These bubbles are NOPs that do not increment PC). In cases of jumps and calls, this logic should flush the pipeline.

f. Data forwarding logic
   Logic is added to provide WB data one clock in advance to RR as needed.

g. Register logic
   Logic to allow simultaneous read of multiple registers in RR and multiple writes in WB is needed.

   It is assumed that 8-bit and 16-bit values will be located in the lowest bits of any register.

h. Memory interface
   Logic to allow arbitration of attempts to read in IF and FO and write in WB must be provided. All signals in the spreadsheet must be implemented for this module.

   The block should support 8-, 16-, and 32-bit operations including shifting to the correct data lines, and bank enable generation (writing), control logic signals (read/write), and wait state generation. It is important that the 8-bit and 16-bit memory are aligned with the least significant bits of the data signals to and from the pipeline, so shifting of the bits is necessary if data from memory r/w operation does not start with bank 0. When reading an 8- or 16-bit value from memory, the upper 24- or 16-bits stored in the register will be sign- or zero-extended.

It is requested that your report should organize information into these 8 blocks. Each block should include a schematic, theory of operation, summary of inputs, outputs, pass-through data, and equations for any logic in the block.

4 Deadlines

Project is due at the date shown in the syllabus, with an oral defense of a written report (containing theory of operation). You may work in teams of one or two members. All members of the team shall participate equally and should be prepared to answer any question about the project to avoid a deduction in points. Each member of the team will be graded independently, although only one report is needed. The projects will not be returned, so please keep a personal copy of the report.

Have fun!