*ECE/CS 552: INTRODUCTION TO COMPUTER ARCHITECTURE*

# Project Description – Stage 1

First Stage Project Report Due: **November 24, 2010**
Final Stage Project Report Due: **December 10, 2010**
Project Demonstrations: **December 12, 2010**

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

**Note**: (rx) stands for the content of register rx.

## 1. WISC-F10 ISA Specifications

WISC-F10 is a 16-bit computer with a load/store architecture. Design and implement this architecture using Altera Quartus II.

WISC-F10 has a register file, a 3-bit FLAG register, and sixteen instructions. The register file comprises sixteen 16-bit registers. Register $0 is hardwired to 0x0000. Register $15 has a special purpose. Register $15 serves as a Return Address register for JAL instruction. The FLAG register contains three bits: Zero (Z), Overflow (V), and Sign bit (N).

WISC-F10's instructions can be categorized into four major classes: Arithmetic, Memory, Load Immediate and Control.

### 1.1 Arithmetic Instructions

Eight arithmetic and logical instructions belong to this category. They are ADD, SUB, AND, OR, SLL, SRL, SRA, and RL.

The ADD, SUB, AND, and OR instructions have a three address format. The assembly level syntax for these instructions is

Opcode rd, rs, rt

The two operands are (rs) and (rt) and the destination is register rd.

The ADD and SUB instructions respectively add and subtract the two operands in two's-complement representation and save the result in register rd.

The AND and OR instructions respectively perform bitwise AND, and bitwise OR operation on the two operands and save the result in register rd.

The SLL, SRL, SRA, and RL instructions have the following assembly level syntax.

Opcode rd, rs, imm

The imm field is a 4-bit immediate operand in unsigned representation for the SLL, SRL, SRA, and RL instructions.

SLL, SRL, SRA and RL shift (rs) by number of bits specified in the imm field and saves the result in register rd. SRA is shift right arithmetic, SRL is shift right logical, SLL is shift left logical, and RL is rotate left.

The machine level encoding for the eight arithmetic/logic instructions is

0aaa dddd ssss tttt

where aaa represents the opcode (see Table 2), dddd and ssss respectively represent the rd and rs registers. The tttt field represents either the rt register or the imm field.

### 1.2 Memory Instructions

There are two instructions which belong to this category: LW, SW. The assembly level syntax for the LW and SW instructions is

Opcode rt, rs, offset

The LW instruction loads register rt with contents from the memory location specified by register rs plus the immediate offset. The signed value offset is sign-extended and added to the contents of register rs to compute the address of the memory location to load.

The SW instruction saves (rt) to the location specified by the register rs plus the immediate offset. The address of the memory location is computed as in LW.

The machine level encoding of these two instructions is

10aa tttt ssss oooo

where aa specifies the opcode, tttt identifies rt, ssss identifies rs, and oooo is the offset in 2's complement representation.

### 1.3 Load Immediate Instruction

There are two instructions which belong to this category: LHB, LLB. The assembly level syntax for the LHB and LLB instructions is

LHB instruction loads the most significant 8 bits of register rt with the bits in the immediate field. The least significant 8 bits of the register rt are left unchanged.

The LLB instruction loads register rt with the 16-bit signed value obtained by sign-extending the 8 bits in the immediate field. The assembly level syntax for LHB and LLB instructions is

Opcode rt, immediate

The machine level encoding for these instructions is

10aa tttt uuuu uuuu

where aa, tttt, and uuuuuuuu respectively specify the opcode, register rt and the 8-bit immediate value.

### 1.4 Control Instructions

There are four instructions which belong to this category: B, JAL, JR, and EXEC.

The B (Branch) instruction conditionally jumps to the address obtained by adding the 8-bit immediate (signed) offset to the contents of the program counter. Assume that the value of the program counter used in this addition is the address of the next instruction (i.e., address of Branch instruction + 1).

The eight possible conditions are Equal (EQ), Not Equal (NE), Greater Than (GT), Less Than (LT), Greater or Equal (GEQ), Less or Equal, Overflow, and True. Many of these conditions are determined based on the 3-bit flag N, V, and Z which should be set by an ADD, SUB, AND or OR instruction executed prior to the conditional branch instruction.

The True condition corresponds to an unconditional branch. The status of the condition is obtained from the FLAG register (definition of each flag is in section 3.3). The assembly level syntax for this instruction is

B cond, offset

The machine level encoding for this instruction is

Opcode 0ccc iiii iiii

where ccc specifies the condition as in Table 1 and iiiiiiii represents the 8-bit signed offset in twos-complement representation.

Table 1: Encoding for Branch conditions

| ccc | *Condition* |
|---|---|
| 000 | Equal (Z = 1) |
| 001 | Not Equal (Z = 0) |
| 010 | Greater Than (Z = N = 0) |
| 011 | Less Than (N = 1) |
| 100 | Greater or Equal (Z = 1 or Z = N = 0) |
| 101 | Less or Equal ( N = 1 or Z = 1) |
| 110 | Overflow (V = 1) |
| 111 | True |

The JAL (jump and link) instruction saves the contents of the program counter (address of the Call instruction + 1) to the Return Address register ($15) and jumps to the procedure whose starting address is partly specified in the instruction. The assembly level syntax for this instruction is

JAL target

The machine level encoding for this instruction is

Opcode gggg gggg gggg

where gggg gggg gggg specifies a 12-bit signed immediate offset that is added to the current PC (which is PC+1 relative to the branch instruction PC) to form the target jump address.

The JR (jump register) instruction jumps to the address specified by (rt). When rt == $15, JR is used as return-from-procedure-call instruction. The assembly level syntax for this instruction is

JR rt

The machine level encoding for this instruction is (tttt encodes the name of register rt)

Opcode tttt xxxx xxxx

The EXEC (execute) instruction jumps to the address specified by (rt), and executes a single instruction from that location in memory, then immediately resumes execution at the instruction following the EXEC instruction (PC+1). If the instruction at the target changes the PC (i.e. it is also a control instruction), that change is ignored. However, if it is any other instruction type it must execute correctly. The assembly level syntax for this instruction is

EXEC rt

The machine level encoding for this instruction is

Opcode tttt xxxx xxxx

Table 2: 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 |

# 2. Memory System

The address space of the WISC-F10 processor is 16 bits. Each memory location is 16 bits in size (word-addressable), so the total memory capacity is $2^{16}$x16 bits = $2^{20}$ bits = 1 Mbit = 128 Kbytes.

For the first stage of the project, the processor will have separate instruction and data memories. The instruction memory has a 16-bit address input and a 16-bit data output with no control signals, while the data memory has a 16-bit address input, a 16-bit data input, a 16-bit data output, and an active-low write signal for specifying a write operation. If the write signal is low, the memory will write the data input bits to the address specified by the address. The read operation is always active for both instruction and data memories and will return data from the location specified by the address. The access time for both memories is one cycle.

Verilog modules will be provided for both memories.

For the second stage of the project, the instruction and data memories will be replaced with cache memories backed up by a slower main memory. **Details of the timing and information of the cache and memory system will be announced later this semester.**

# 3. Implementation

## 3.1 Pipelining

Your design must use a five stage pipeline (IF, ID, EX, MEM, WB) similar to that in the text. The design should follow the Quartus II design problems presented in the homework assignments during the course of the semester. You must implement hazard detection so that your pipeline correctly handles all data and control dependences.

## 3.2 Reset Sequence

WISC-F09 has a Reset input. Instructions are executed when Reset input is low. If the Reset input goes high for one clock cycle, the contents of the program counter and register file are cleared.

### 3.3 Flags

Flag bits are stored in the FLAG register and are used in conditional jump. There are three bits in the FLAG register: Zero (Z), Overflow (V), and Sign bit (N). Only four instructions can change the FLAG register: ADD, SUB, AND, OR.

The Z flag is set if and only if the output of the operation is zero.

The V flag is set by the ADD and SUB instructions if and only if the operation results in an overflow. Overflow must be set based on treating the arithmetic values as 16-bit signed integers. The AND and OR instructions always clear the V flag.

The N flag is set if and only if the result of the ADD and SUB instruction is negative. **Note that N flag should be implemented as (sign-bit XOR overflow).** The AND and OR instructions always clear the N flag.

Other Instructions, including shift/rotate instructions, load/store instructions and control instructions, do not change the contents of the FLAG register.

### 3.4 Extra features

You can get bonus points by implementing an interesting feature that is not part of this specification. For example, use of data forwarding is an additional feature. Implementation of a non-trivial branch predictor is an additional feature. Implementation of exception handling is also an additional feature.

You will get bonus points only if you have a completely working design meeting all the specifications in this document. Remember to save the working design before you begin work on the extra feature.

Implementing additional interesting instructions will also be considered an extra feature. If you implement an additional instruction, clearly specify its assembly level syntax and machine level encoding. Also fully define the actions carried out by the instruction. Write a small program to illustrate the potential improvement in runtime that can result of including your new instruction. Since the opcode space is fully utilized you will have to omit existing instructions to make room for new ones. Make sure you maintain a fully-working implementation for the required instruction set in parallel with your modified instruction set. Some possibilities here include XOR, population count, 16-bit floating-point multiply and/or add.

**Make sure you discuss any extra features with the TA and instructor before proceeding.**

## 4. Submission Requirements

### First Stage Project Report

The project should be done in a group of three. The first stage project report should include the following parts:

1) A brief description of or an introduction to your project. 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.

4) The simulation results for test programs (will be provided later) in "list" form.
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. A significant part of your project grade will be based on the testing methodology you used.
6) If your project implements extra features (forwarding, etc.) to further enhance performance, 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.

Note that there will be additional documentation requirements for the stage 2 memory system design which will be specified later.

## Project Demonstration

Each group will be asked to demonstrate their project on **Sunday December 12, 2010**. All team members must be present at the demo and must be prepared to answer detailed questions about the design. The demonstration will be used to primarily evaluate the following.

- Verification methodology: Designs tested with well thought out systematic procedures will get higher scores. You will be asked to show the test programs you have used to create test inputs and/or verify the outputs from the various blocks.