ECE/CS 552: INTRODUCTION TO COMPUTER ARCHITECTURE

Project Description – Stage 1

Due on Oct. 26th.

1. WISC-F15 ISA Specifications

WISC-F15 is a 16-bit computer with load/store architecture. Design, and test this architecture using Verilog.

WISC-F15 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 serves as a Return Address register for the CALL instruction. The FLAG register contains three bits: Zero (Z), Overflow (V), and Sign bit (N).

WISC-F15'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, PADDSB, SUB, NAND, XOR, SLL, SRL, SRA.

The ADD, PADDSB, SUB, NAND, and XOR, instructions have a three-address format. The assembly level syntax for these instructions is

Opcode rd, rs, rt

The two operands are[1] (rs) and (rt) and the destination is register rd.

The ADD, PADDSB, and SUB instructions respectively add, parallel add split bytes, and subtract the two operands (rs – rt) in two's-complement representation and save the result in register rd.

The ADD and SUB instructions will use saturating arithmetic. Meaning if a result exceeds the most positive number (2^15–1) then the result is saturated to (2^15–1). Likewise if the result were to underflow the most negative number (–2^15) then the result would be saturated to –2^15.

The PADDSB instruction performs two byte-wise additions in parallel to realize *sub-word parallelism*. Specifically, the higher byte and the lower byte will be treated as two separate numbers stored in a single word as a byte vector. When PADDSB is performed, the two numbers will be added separately. To be more specific, let the contents in rs and rt are aaaa_bbbb_cccc_dddd, eeee_ffff_gggg_hhhh respectively where a, b, c, d, e, f, g, and h $\in$ {0, 1}. Then after execution of PADDSB, the content of rd will be {saturation of (aaaa_bbbb+eeee_ffff), saturation of (cccc_dddd+gggg_hhhh)}. =====The two bytes of result should be saturated separately, meaning if a result exceeds the most positive number (2^7-1) then the result is saturated to (2^7-1), and if the result were to underflow the most negative number (-2^7) then the result would be saturated to -2^7.

The NAND, and XOR instructions respectively perform bitwise NAND, and bitwise XOR, operations on the two operands and save the result in register rd.

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

[1] (rx) stands for the content of register rx.

Opcode rd, rs, imm

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

SLL, SRL, and SRA 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, finally SLL is shift left logical..

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: LLB Rx, 0xYY

LHB Rx, 0xYY

Where Rx is the register being loaded, and 0xYY is the 8-bit immediate value. For LLB the 8-bit immediate value will be sign extended and loaded into Rx.

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

The machine level encoding for these instructions is

101a dddd uuuu uuuu

where a is a bit to differentiate LLB vs LHB, dddd specifies the destination registers, and uuuuuuuu is the 8-bit immediate value.

1.4 Control Instructions

There are four instructions which belong to this category: B, CALL, RET, and HLT

The B (Branch) instruction conditionally jumps to the address obtained by adding the 9-bit immediate (signed) offset to the contents of the program counter+1. 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 (NEQ), Greater Than (GT), Less Than (LT), Greater Than or Equal (GTE), Less Than or Equal (LTE), Overflow (OVFL), and Unconditional (UNCOND). Many of these conditions are determined based on the 3-bit flag N, V, and Z. Instruction that set flags are outlined in the table below:

Function	Flags Set
ADD	N,Z,V
SUB	N,Z,V
NAND	Z
XOR	Z
SLL	Z
SRL	Z
SRA	Z

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, Label

The machine level encoding for this instruction is

Opcode ccci iiii iiii

where ccc specifies the condition as in Table 1 and iiiiiiiii represents the 9-bit signed offset in twos-complement representation.

Table 1: Encoding for Branch conditions

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

The CALL 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

CALL 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 PC of the CALL instruction) to form the target jump address.

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

RET rs

The machine level encoding for this instruction is (ssss encodes the name of register rs)

Opcode xxxx tttt xxxx

The HLT instruction essentially freezes the whole machine. PC stops advancing and the pipe is stalled.

Table 2: Table of opcodes

Function	Opcode
ADD	0000
PADDSB	0001
SUB	0010
NAND	0011
XOR	0100
SLL	0101
SRL	0110
SRA	0111
LW	1000
SW	1001
LHB	1010
LLB	1011
B	1100
CALL	1101
RET	1110
HLT	1111

2. Memory System

The address space of the WISC-F15 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, a read enable, and a 16-bit data output. The data memory has a 16-bit address input, a 16-bit data input, a 16-bit data output, a write enable signal, and a read enable signal (both read and write cannot be active in the same clock cycle). If the write signal is asserted, the memory will write the data input bits to the address specified by the address. The read operation is active when the read enable signal is high. Both instruction and data memories perform their operations in a single clock 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 in the second stage description of this project.**

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 will make use of Verilog modules you developed as part of 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-F15 has an active low reset input (rst_n). Instructions are executed when rst_n is high. If rst_n goes low for one clock cycle, the contents of the state of the machine is reset and starts execution at address 0x0000.

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 the arithmetic instructions (except PADDSB) can change all three flags (Z,OV,N). The logical instructions (AND, XOR, SLL, SRL, SRA) can change the Z FLAG.

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 N flag is set if and only if the result of the ADD, and SUB instruction is negative.

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

4. Interface

Your top level verilog should be file called ***cpu.v***. It should have a simple 3 signal interface: ***clk***, ***rst_n***, and ***hlt***.

Signal Interface of ***cpu.v***		
Signal:	**Direction:**	**Description:**
clk	in	System clock
rst_n	in	Active low reset. A low on this signal resets the processor and causes execution to start at address 0x0000
hlt	out	When your processor encounters the HLT instruction it will assert this signal once it is finished processing the instruction prior to the HLT. This signal should also be connected to the ***hlt*** input of the RF, which causes the RF to dump its register contents at the end of the test.
pc[15:0]	Out	The RoadRunner script used to check the project results will compare the PC to the correct value for the PC (one instruction after the HLT instruction)

5. Submission Requirements

First stage project submission will be done as a zip file, and uploaded to the dropbox on moodle. After unzip the zip file, there should be the following files:

Project report, which states what you did in your project, and how to run your code using your testbench. The project report should also include the screen shot of the testing results, one or more for each test.

Project report sheet, which will be posted on moodle.

A directory, which contains all the Verilog, files of your design, including testbench you used.

Another directory, which includes all the binary test files.