

COE 405
Design & Modeling of Digital Systems

Course Project#2 – Term 122
RISC Processor Design
Due on: Wednesday May 15, 2013

In this project, you will design and synthesize a 16-bit MIPS-like processor. The details about the processor are given below.

Instruction Set Architecture

The 16-bit MIPS-like processor has seven 16-bit general-purpose registers: R1 through R7. R0 is hardwired to zero and cannot be written. There is also one special-purpose 12-bit register, which is the program counter (PC). All instructions are 16 bits and there are three instruction formats: R-type, I-type, and J-type as shown below:

R-type format

4-bit opcode (Op), 3-bit register numbers (Rs, Rt, and Rd), and 3-bit function field (funct)

Op ⁴	Rs ³	Rt ³	Rd ³	funct ³
-----------------	-----------------	-----------------	-----------------	--------------------

I-type format

4-bit opcode (Op), 3-bit register numbers (Rs and Rt), and 6-bit immediate constant

Op ⁴	Rs ³	Rt ³	Immediate ⁶
-----------------	-----------------	-----------------	------------------------

J-type format

4-bit opcode (Op) and 12-bit immediate constant

Op ⁴	Immediate ¹²
-----------------	-------------------------

For R-type instructions, Rs and Rt specify the two source register numbers, and Rd specifies the destination register number. The function field can specify at most eight functions for a given opcode. Opcodes 0 and 1 are reserved for R-type instructions.

For I-type instructions, Rs specifies a source register number, and Rt can be a second source or a destination register number. The immediate constant is only 6 bits because of the fixed-size nature of the instruction. The 6-bit immediate constant is assumed to be sign-extended for all instructions.

For J-type, a 12-bit immediate constant is used for J (jump), JAL (jump-and-link), and LUI (load upper immediate) instructions.

Instruction Encoding

Sixteen R-type instructions, eleven I-type instructions, and three J-type instructions are defined. These instructions, their meaning, and their encoding are shown below:

Instr	Meaning	Encoding				
		Op	Rs	Rt	Rd	f
AND	$\text{Reg(Rd)} = \text{Reg(Rs)} \& \text{Reg(Rt)}$	Op = 0000	Rs	Rt	Rd	f = 000
OR	$\text{Reg(Rd)} = \text{Reg(Rs)} \text{Reg(Rt)}$	Op = 0000	Rs	Rt	Rd	f = 001
NOR	$\text{Reg(Rd)} = \sim(\text{Reg(Rs)} \text{Reg(Rt)})$	Op = 0000	Rs	Rt	Rd	f = 010
XOR	$\text{Reg(Rd)} = \text{Reg(Rs)} \wedge \text{Reg(Rt)}$	Op = 0000	Rs	Rt	Rd	f = 011
SLL	$\text{Reg(Rd)} = \text{Reg(Rs)} \ll \text{Reg(Rt)}$	Op = 0000	Rs	Rt	Rd	f = 100
SRL	$\text{Reg(Rd)} = \text{Reg(Rs)} \text{ zero} \gg \text{Reg(Rt)}$	Op = 0000	Rs	Rt	Rd	f = 101
SRA	$\text{Reg(Rd)} = \text{Reg(Rs)} \text{ sign} \gg \text{Reg(Rt)}$	Op = 0000	Rs	Rt	Rd	f = 110
ROL	$\text{Reg(Rd)} = \text{Reg(Rs)} \text{ rotate} \ll \text{Reg(Rt)}$	Op = 0000	Rs	Rt	Rd	f = 111
ADD	$\text{Reg(Rd)} = \text{Reg(Rs)} + \text{Reg(Rt)}$	Op = 0001	Rs	Rt	Rd	f = 000
SUB	$\text{Reg(Rd)} = \text{Reg(Rs)} - \text{Reg(Rt)}$	Op = 0001	Rs	Rt	Rd	f = 001
SLT	$\text{Reg(Rd)} = \text{Reg(Rs)} \text{ signed} < \text{Reg(Rt)}$	Op = 0001	Rs	Rt	Rd	f = 010
SLTU	$\text{Reg(Rd)} = \text{Reg(Rs)} \text{ unsigned} < \text{Reg(Rt)}$	Op = 0001	Rs	Rt	Rd	f = 011
DIV	$\text{Reg(Rd)} = \text{Quot}(\text{Reg(Rs)} / \text{Reg(Rt)})$	Op = 0001	Rs	Rt	Rd	f = 100
REM	$\text{Reg(Rd)} = \text{Rem}(\text{Reg(Rs)} / \text{Reg(Rt)})$	Op = 0001	Rs	Rt	Rd	f = 101
MUL	$\text{Reg(Rd)} = \text{Reg(Rs)} * \text{Reg(Rt)}$	Op = 0001	Rs	Rt	Rd	f = 110
JR	PC = lower 12 bits of Reg(Rs)	Op = 0001	Rs	000	000	f = 111
LW	$\text{Reg(Rt)} = \text{Mem}(\text{Reg(Rs)} + \text{ext}(\text{im}^6))$	Op = 0010	Rs	Rt	Immediate ⁶	
SW	$\text{Mem}(\text{Reg(Rs)} + \text{ext}(\text{im}^6)) = \text{Reg(Rt)}$	Op = 0011	Rs	Rt	Immediate ⁶	
ANDI	$\text{Reg(Rt)} = \text{Reg(Rs)} \& \text{ext}(\text{im}^6)$	Op = 0110	Rs	Rt	Immediate ⁶	
ORI	$\text{Reg(Rt)} = \text{Reg(Rs)} \text{ext}(\text{im}^6)$	Op = 0111	Rs	Rt	Immediate ⁶	
ADDI	$\text{Reg(Rt)} = \text{Reg(Rs)} + \text{ext}(\text{im}^6)$	Op = 1000	Rs	Rt	Immediate ⁶	
BEQ	Branch if $(\text{Reg(Rs)} == \text{Reg(Rt)})$	Op = 0100	Rs	Rt	Immediate ⁶	
BNE	Branch if $(\text{Reg(Rs)} != \text{Reg(Rt)})$	Op = 0101	Rs	Rt	Immediate ⁶	
BLTZ	Branch if $(\text{Reg(Rs)} < 0)$	Op = 1100	Rs	Rt	Immediate ⁶	
BLEZ	Branch if $(\text{Reg(Rs)} \leq 0)$	Op = 1101	Rs	Rt	Immediate ⁶	
BGTZ	Branch if $(\text{Reg(Rs)} > 0)$	Op = 1110	Rs	Rt	Immediate ⁶	
BGEZ	Branch if $(\text{Reg(Rs)} \geq 0)$	Op = 1111	Rs	Rt	Immediate ⁶	
J	PC = Immediate ¹²	Op = 1001	Immediate ¹²			
JAL	R7 = PC + 1, PC = Immediate ¹²	Op = 1011	Immediate ¹²			
LUI	R1 = Immediate ¹² \ll 4	Op = 1010	Immediate ¹²			

There are three shift and one rotate instructions. For shift and rotate instructions, the least significant 4 bits of register Rt are used as the shift/rotate amount. There is only one rotate left (ROL) instruction. To rotate right by n bits, you can rotate left by $16 - n$ bits, because registers are 16 bits. The Load Upper Immediate (LUI) is of the J-type to have a 12-bit immediate constant loaded into the upper 12 bits of register R1. The LUI can be combined with ORI (or ADDI) to load any 16-bit constant into a register. Although the instruction set is reduced, it is still rich enough to write useful programs. We can have procedure calls and returns using the JAL and JR instructions.

Memory

Your processor will have separate instruction and data memories with $2^{12} = 4096$ words each. Each word is 16 bits or 2 bytes. Memory is *word addressable*. Only words (not bytes) can be

read and written to memory, and each address is a word address. This will simplify the processor implementation. The PC contains a word address (not a byte address). Therefore, it is sufficient to increment the PC by 1 (rather than 2) to point to the next instruction in memory. Also, the Load and Store instructions can only load and store words. There is no instruction to load or store a byte in memory.

Addressing Modes

For branch instructions (BEQ, BNE, BLTZ, BLEZ, BGTZ and BGEZ), PC-relative addressing mode is used: $PC = PC + \text{sign-extend}(\text{immediate})^6$. For jump instructions (J and JAL), direct addressing is used: $PC = \text{Immediate}^{12}$. For LW and SW instructions, base-displacement addressing mode is used. The base address in register Rs is added to the sign-extended immediate⁶ to compute the memory address.

Program Execution

The program will be loaded and will start at address 0 in the instruction memory. The data segment will be loaded and will start also at address 0 in the data memory. To terminate the execution of a program, the last instruction in the program can jump or branch to itself indefinitely.

Testing

To test your CPU, implement any sort procedure of your choice and use this procedure to sort an array of 8 words of your choice. Display the memory content of these words to be sorted on the LCD screen and control the speed of the CPU having four different speeds of the clock: 1HZ, 10HZ, 50HZ and 100HZ.

Project Report

The report document must contain sections highlighting the following:

1 – Design and Implementation

- Specify clearly the data path design
- Specify the control unit using ASMD chart.
- Carry out the design and implementation with the following aspects in mind:
 - Correctness of the individual components
 - Correctness of the overall design when wiring the components together
 - Completeness: all instructions were implemented properly

2 – Simulation and Testing

- Carry out the simulation of the processor developed using Modelsim.
- Test each of the components individually and demonstrate its correct operation including the ALU and register file.
- Describe the test programs that you used to test your design with enough comments describing the program, its inputs, and its expected output. List all the instructions that were tested and work correctly. List all the instructions that do not run properly.
- Also provide snapshots of the Simulator window with your test program loaded and showing the simulation output results.
- Synthesize the processor on FPGA and demonstrate its correct functionality by sorting an array of data displayed on LCD.

3 – Teamwork

- This project is a team work project with two students per team. Make sure to write the names of all the group members on the project report title page.
- Each group should assign a group leader that leads the conduction of the project, divide the project tasks among the team members.
- Project tasks should be divided among the group members so that each group member contributes equally in the project and everyone is involved in all the following activities:
 - Design and Implementation
 - Simulation and Testing
 - Synthesis and FPGA implementation
 - Design and results reporting
- Clearly show the work done by each group member
- Students who **help** other team members should mention that to earn credit for that.

Submission Guidelines

All submissions will be done through WebCT.

Attach one zip file containing all verilog and constraint files used in your design, a video demo of your project, as well as the report document.

Submit also a hard copy of the report during the class lecture.

Grading policy:

Grading Criteria	Mark
Demonstration of correct functionality of components and whole design by simulation	50
Demonstration of correct functionality of project on FPGA	30
Project Documentation and Report Organization	20
Total	100