BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering
Second Semester, 2020/2021

Project -2-
Computer Architecture - ENCS4370
Due Tuesday, May 25, 2021

---

1. Objectives:

- Using the Logisim simulator
- Designing and testing a Pipelined 16-bit processor

2. Instruction Set Architecture

In this project, you will design a simple 16-bit RISC processor with eight 16-bit general-purpose registers: R0 through R7. R0 is hardwired to zero and cannot be written, so we are left with seven registers. There is also one special-purpose 16-bit register, which is the program counter (PC). All instructions are 16 bits. There are three instruction formats (R-type, I-type, and J-type) and five addressing modes. The processor has a memory with word size equal 16-bit and a total size of 2^{16} bytes.

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 number (Rs and Rt), and 6-bit signed 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. We will reserve opcode 0 for R-type instructions. It is also possible to reserve more opcodes, if more R-type instructions exist.

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

size of the immediate constant is suitable for our uses. The 6-bit immediate constant is signed (and sign-extended) for all I-type instructions.

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

3. Instruction Encoding

8 R-type instructions, 9 I-type instructions, and three J-type instructions are defined. These instructions, their meaning, and their encoding are shown below:

Instr	Meaning	Encoding				
AND	Reg(Rd) = Reg(Rs) & Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 111
OR	Reg(Rd) = Reg(Rs) | Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 110
CAS	Reg(Rd) = Max[Reg(Rs) , Reg(Rt)]	Op = 0000	Rs	Rt	Rd	f = 101
Lws	Reg(Rd) = Mem[Reg(Rs) + Reg(Rt)]	Op = 0000	Rs	Rt	Rd	f = 100
ADD	Reg(Rd) = Reg(Rs) + Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 011
SUB	Reg(Rd) = Reg(Rs) – Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 010
SLT	Reg(Rd) = Reg(Rs) < Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 001
JR	PC = Reg(Rs)	Op = 0000	Rs	000	000	f = 000
ANDI	Reg(Rt) = Reg(Rs) & Immediate^6	Op = 0001	Rs	Rt	Immediate^6	
ORI	Reg(Rt) = Reg(Rs) | Immediate^6	Op = 0010	Rs	Rt	Immediate^6	
ADDI	Reg(Rt) = Reg(Rs) + Immediate^6	Op = 0011	Rs	Rt	Immediate^6	
Lw	Reg(Rt) = Mem(Reg(Rs) + Imm^6)	Op = 0100	Rs	Rt	Immediate^6	
Lbu	Reg(Rt [7:0]) = Mem(Reg(Rs) + Imm^6) Reg(Rt [15:8]) = 00000000	Op = 0101			Immediate^6	
Lb	Reg(Rt [7:0]) = Mem(Reg(Rs) + Imm^6) Reg(Rt [15:8]) = sign bit	Op = 0110			Immediate^6	
Sw	Mem(Reg(Rs) + Imm^6) = Reg(Rt)	Op = 0111	Rs	Rt	Immediate^6	
Sb	Mem(Reg(Rs) + Imm^6) = Reg(Rt [7:0])	Op = 1000			Immediate^6	
BEQ	Branch if (Reg(Rs) == Reg(Rt))	Op = 1001	Rs	Rt	Immediate^6	
J	PC = PC + Immediate	Op = 1010	Immediate^{12}			
JAL	R7 = PC + 1, PC = PC + Immediate	Op = 1011	Immediate^{12}			
LUI	R1 = Immediate^{12} << 4	Op = 1100	Immediate^{12}			

Opcode 0 is used for R-type instructions. 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.

4. Getting Started with Logisim

You should first download Logisim from the Logisim website http://ozark.hendrix.edu/~burch/logisim/. Logisim is very easy to use. To get started, you can read the documentation available under the Logisim website, Or from http://www.youtube.com/watch?v=ATPqpFMlVdw as an examples.

WARNING: Although Logisim is stable, it might crash from time to time. Therefore, it is best to save your work often. Make several copies and versions of your design before making changes, in case you need to go back to an older version.

5. Building a Pipelined Processor

Design and implement a pipelined-datapath and its control logic. A five-stage pipeline should be constructed similar to the pipeline presented in the class lectures. Add pipeline registers between stages. Design the control logic to detect data dependencies among instructions and implement the forwarding logic. For branch and jump instructions, reduce the delay to one cycle only. Stall the pipeline for one clock cycle after a jump or a taken branch instruction. If the branch is not taken, then there is no need to stall the pipeline. You should design the datapath to support the above instructions with minimum stall cycles.

6. Testing

To test the implementation, write a sample code to test all the instructions that you have implemented. Note that, 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.

7. Project Report

The report document must contain sections highlighting the following:

1 – Design and Implementation

- Specify clearly the design giving detailed description of the datapath, its components, control, and the implementation details (highlighting the design choices you made and why, and any notable features that your processor might have.)
- Provide drawings of the component circuits and the overall datapath.
- Provide a complete description of the control logic and the control signals. Provide a table giving the control signal values for each instruction. Provide the logic equations for each control signal.
- Provide a complete description of the forwarding logic, the cases that were handled, and the cases that stall the pipeline, and the logic that you have implemented to stall the pipeline.
- Provide list of sources for any parts of your design that are not entirely yours (if any).
- 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, detecting dependences and forwarding was handled properly, and stalling the pipeline was handled properly for all cases.

2 – Simulation and Testing

- Carry out the simulation of the processor developed using Logisim.
- 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.
- Describe all the case that you handled involving dependences between instructions, forwarding cases, and cases that stall the pipeline.
- Also provide snapshots of the Simulator window with your test program loaded and showing the simulation output results.

3 – Teamwork

- At most three students can form a group.
- Group members are required to coordinate the work equally among themselves so that everyone is involved in all the following activities:
  - Design and Implementation
  - Simulation and Testing
- Clearly show the work done by each group member.

8. Submission Guidelines

Attach one zip file containing all the design circuits, the programs source code and binary instruction files that you have used to test your design, their test data, as well as the report document to ritaj.

9. Grading policy

The grade will be divided according to the following components:

- Correctness: whether your implementation is working
- Completeness and testing: whether all instructions and cases have been implemented, handled, and tested properly
- Participation and contribution to the project
- Report document
- Discussion