COE 308 – Computer Architecture
Term 061 – Fall 2006

Project 3: Pipelined Processor Design

Due Sunday, January 14, 2006 by Midnight

Objectives:

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

Instruction Set Architecture

In this project, you will use the same 16-bit MIPS-like instruction set that was defined in the previous project. There are seven 16-bit general-purpose registers: R1 through R7. R0 is hardwired to zero and cannot be written. There is also one special-purpose 16-bit register, which is the program counter (PC). There are three instruction formats 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 number (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	$Immediate^{12}$

Instruction Encoding:

Eight R-type instructions, six I-type instructions, and three J-type instructions are defined. These instructions, their meanings, and their encodings are shown below:

Instr	Meaning	Encoding				
OR	Reg(Rd) = Reg(Rs) | Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 000
AND	Reg(Rd) = Reg(Rs) & Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 001
NOR	Reg(Rd) = ~(Reg(Rs) | Reg(Rt))	Op = 0000	Rs	Rt	Rd	f = 010
XOR	Reg(Rd) = Reg(Rs) ^ Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 011
ADD	Reg(Rd) = Reg(Rs) + Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 100
SUB	Reg(Rd) = Reg(Rs) – Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 101
SLT	Reg(Rd) = Reg(Rs) < Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 110
JR	Jump register: PC = Reg(Rs)	Op = 0000	Rs	000	000	f = 111

ADDI	Reg(Rt) = Reg(Rs) + ext(im^6)	Op = 0100	Rs	Rt	$Immediate^6$
SLTI	Reg(Rt) = Reg(Rs) < ext(im^6)	Op = 0110	Rs	Rt	$Immediate^6$
LW	Reg(Rt) = Mem(Reg(Rs) + ext(im^6))	Op = 1000	Rs	Rt	$Immediate^6$
SW	Mem(Reg(Rs) + ext(im^6)) = Reg(Rt)	Op = 1001	Rs	Rt	$Immediate^6$
BEQ	Branch if (Reg(Rs) == Reg(Rt))	Op = 1010	Rs	Rt	$Immediate^6$
BNE	Branch if (Reg(Rs) != Reg(Rt))	Op = 1011	Rs	Rt	$Immediate^6$
J	PC = PC + 1 + ext(im^{12})	Op = 1100	$Immediate^{12}$		
JAL	R7 = PC + 1, PC = PC+1+ext(im^{12})	Op = 1101	$Immediate^{12}$		
LUI	R1 = $Immediate^{12}$ << 4	Op = 1111	$Immediate^{12}$		

Memory:

Your processor will have separate instruction and data memories as in the previous project. Memory is *word addressable*. Each word is 16 bits or 2 bytes.

Addressing Modes:

For branches (BEQ and BNE) and jumps (J and JAL), PC-relative addressing mode is used. PC = PC + 1 + sign-extend(imm^6) for branches and PC = PC + 1 + sign-extend(imm^{12}) for jumps. For LW and SW base-displacement addressing mode is used. The base address in register Rs is added to the sign-extended $immediate^6$ 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. You may also have a stack segment if you want to support procedures. The stack segment can occupy the upper part of the data memory and can grow backwards towards lower addresses. The stack segment can be implemented completely in software. To terminate the execution of a program, the last instruction in the program can jump or branch to itself indefinitely.

Building a Pipelined Processor

In this project, you will build a 5-stage pipelined datapath and its control logic. You may start with the non-pipelined single cycle processor that you have implemented in the first project and modify it to make it pipelined. **You should detect data hazards, implement forwarding, detect and handle control hazards, and stall the pipeline when necessary.** You can test this part by loading a program into the instruction memory and loading its data into the data memory.

Testing:

To test the implementation, write simple programs to test whether all instructions are implemented and pipelined correctly. You should also test whether data hazards are detected properly, whether forwarding is implemented properly, and whether control hazards are detected and handled properly.

Groups:

As in the previous project, two or at most three students can form a group. It is best to continue with the same group. Make sure to write the names of all the students involved in your group on the project report.

Submission Guidelines:

All submissions will be by email sent to:

mudawar@ccse.kfupm.edu.sa ; rfarooqi@ccse.kfupm.edu.sa

Subject: COE 308 Project 3

Attach one zip file containing the design circuits, the binary image files of the sample programs that you have used to test your design, as well as the report document.

Report:

The report should contain the names of all the group members and the collaboration efforts among the group members (how the group members collaborated and divided the work among themselves). It should contain the circuit diagrams and a description of the circuit design. You should describe the sample code that was used to test your design in assembly language and in binary. Make sure to demonstrate that instructions were implemented correctly. Also, demonstrate pipelining, describe how data hazards are detected, how forwarding is implemented, how control hazards are handled. Make sure to report all cases. You will loose points on the report if the report is not complete.

Grading policy:

The grade will be divided according to the following components:

- Correctness: whether your implementation is working
- Completeness and testing: whether all instructions have been implemented correctly, and all data and control hazards have been tested properly.
- Participation and contribution to the project
- Report document

Late policy:

The project should be submitted on the due date by midnight. Late projects are accepted, but will be penalized 5% for each late day and for a maximum of 5 late days (or 25%). Projects submitted after 5 late days will not be accepted.