Design Document
CS 151/251
Alex & Niv

How many stages is the datapath you've drawn? (i.e. How many cycles does it take to execute 1 instruction?)
3 stages -> 3 cycles

2. How do you handle ALU → ALU hazards?
addi x1, x2, 100
addi x2, x1, 100

We are considering implementing forwarding

3. How do you handle ALU → MEM hazards?
addi x1, x2, 100
sw x1, 0(x3)

We are going to implement forwarding

4. How do you handle MEM → ALU hazards?
lw x1, 0(x3)
addi x1, x1, 100

We are going to implement forwarding

5. How do you handle MEM → MEM hazards?
lw x1, 0(x2)
sw x1, 4(x2)
also consider:
lw x1, 0(x2)
sw x3, 0(x1) 22

Pads with nops

6. Do you need special handling for 2 cycle apart hazards?
addi x1, x2, 100
nop
addi x1, x1, 100

This does not require special handling in our system.

7. How do you handle branch control hazards? (What is the mispredict latency, what prediction scheme are you using, are you just injecting NOPs until the branch is resolved, what about data hazards in the branch?)

We are using a 2 bit BHT that looks at fetch pc's. We are aiming to implement the gshare algorithm.

8. How do you handle jump control hazards? Consider jal and jalr separately. What optimizations can be made to special-case handle jal?

Jal rd, immediate
Rd = PC + 4
PC = PC+offset

JALR rd, rs, immediate
Rd = PC+4
PC = rs + immediate (12 bit, sign extended)

9. What is the most likely critical path in your design?

Reading and write from memory in MW

10. Where do the UART modules, instruction, and cycle counters go? How are you going to drive uart_tx_data_in_valid and uart_rx_data_out_ready (give logic expressions)?

Case 8h'08:
Uart_tx_data_in_valid = opcode == 7'b0100011 ## we know if opcode is OPC_STORE and the address is 0x80..08, then data will be valid so data from CPU goes to UART

case 8h'04:
data_enc = {24'b0, uart_rx_data_out}
Uart_rx_data_out_ready = opcode == 7'b0000011 ## OPC_LOAD, UART to CPU

Table 3: I/O Memory Map

Address	Function	Access	Data Encoding
32'h80000000	UART control	Read	{30'b0, uart_rx_data_out_valid, uart_tx_data_in_ready}
32'h80000004	UART receiver data	Read	{24'b0, uart_rx_data_out}
32'h80000008	UART transmitter data	Write	{24'b0, uart_tx_data_in}
32'h80000010	Cycle counter	Read	Clock cycles elapsed
32'h80000014	Instruction counter	Read	Number of instructions executed
32'h80000018	Reset counters to 0	Write	N/A

11. What is the role of the CSR register? Where does it go?

CSR registers go in stage 3 during Memory WB as you are checking to make sure the behavior and status of your processor is correct.

12. When do we read from BIOS for instructions? When do we read from IMem for instructions? How do we switch from BIOS address space to IMem address space? In which case can we write to IMem, and why do we need to write to IMem? How do we know if a memory instruction is intended for DMem or any IO device?

When we get instructions from IO's like a UART we execute from BIOS memory
IMEM - Programs are downloaded into IMEM for operation, we read from IMEM when we request BIOS to Jump into IMEM address.
Write to IMEM - Store the program for execution
Switch from BIOS to IMEM - Jump instruction with address pointing to IMEM in BIOS code
PC bit 30 tell us if we need to read from BIOS/IMEM
Address bit 31:28 would tell us if the instruction is intended to DMEM or IODevice

Address[31:28]	Address Type	Device	Access	Notes
4'b00x1	Data	Data Memory	Read/Write	
4'b0001	PC	Instruction Memory	Read-only	
4'b001x	Data	Instruction Memory	Write-Only	Only if PC[30] == 1'b1
4'b0100	PC	BIOS Memory	Read-only	
4'b0100	Data	BIOS Memory	Read-only	
4'b1000	Data	I/O	Read/Write	

Commit your block diagram and your writeup to your team repository under sp22_fpga_teamXX/docs by April 4, 2022. Please also remember to push your working IO circuits to your GitHub repository.