Michael Lankford

Homework 6

2/14/2023

1.

* Pipelining is used to achieve a higher clock rate of a multicycle design and a single-cycle CPI of a single-clock design. This is done by overlapping instructions during execution, which reduces the total time to execute them.
* Pipelining in MIPS takes five stages to complete instruction and execution:
  + Instruction Fetch - fetch the instruction from memory
  + Instruction Decode - read registers while decoding the instruction
  + Execution - execute the operation or calculate the address
  + Memory - access an operand in data memory
  + Write Back - write the result into a register
* Pipeline hazards are occurrences where the next instruction cannot execute in the following clock cycle. There are 3 types of these hazards:
  + Structural hazard – occur when the hardware can’t support the combination of instructions that are wanted to execute in the same clock cycle / Ex: 2 instructions try to access the same memory resource at the same time
  + Data hazard – occur when the pipeline has to be stalled because a step must wait for another to complete / Ex: having an add instruction followed by a subtraction instruction
  + Control hazards – occurs when the need to make a decision based on the results of one instruction while others are executing / Ex: branches are the usual causes of this type of hazard
* Forwarding(bypassing) is used to resolve a data hazard by retrieving the missing data rather than waiting for the instruction to complete in the correct pipeline order.
* A load-use data hazard is a form of data hazard where the data being loaded by a load instruction has not yet become available when it’s needed by another instruction.
* Code scheduling is used to reduce or prevent all hazards.
* Branch prediction is used to resolve branch hazards (control hazards) by assuming a given outcome for the branch and proceeding from the previous assumption. There are several types of branch prediction:
  + Dynamic branch prediction – predicting branches at runtime using runtime information
  + Buffer branch prediction – predicts branches at runtime but does not supply the target PC value
* Instruction level parallelism is parallelism among instructions and is the basis of pipelining (running multiple instructions in parallel to decrease runtime).
* Multiple issue is a scheme where multiple instructions are launched in one clock cycle. There are 2 ways to do this:
  + Static multiple issue – many decisions are made by the compiler before execution
  + Dynamic multiple issue – many decisions are made during execution by the processor
* Loop unrolling is a technique to get more performance from loops that access array, where multiple copies of the loop body are made and instructions from different iterations are scheduled together.

2. Exercise 4.3

4.3.1

The fraction of all instructions that use data memory is 35% because the load and store instructions deal with memory, so you’d add up 25% and 10%.

4.3.2

The fraction of all instructions that use instruction memory is 100% because every instruction uses instruction memory.

4.3.3

The fraction of instructions that use the sign extend is 76% because the I-type, Load, Store, Branch, and Jump instructions all use the sign extend. That leaves the R-type, so you subtract 24% from 100% and come out with 76%.

4.3.4

During cycles in which its output is not needed, the sign extend still produces an output, however, nothing will be outputted because it is not needed.

3. Exercise 4.4

4.4.1

The load instruction fails to operate correctly if the MemToReg wire is stuck at 0 because it takes data from memory and moves it to a register.

4.4.2

The load instructions, store instructions, and immediate type instructions break if the ALUSrc wire is stuck at 0.

4. Exercise 4.5

4.5.1

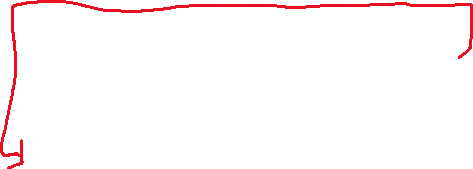
Because the ALU is fetching a word(a.k.a. loading a word), the values for its inputs of the instruction would be

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| opcode | ALUOp | Operation | funct | ALU function | ALU control |
| lw | 00 | Load word | XXXXXX | add | 0010 |

4.5.2

The new PC address is found by taking the old PC address and adding 4 to it. The path taken to get this value is: Diagram

Description automatically generated



4.5.3

PCSrc Mux:

Input: PC+4

Output: PC+0x28

ALUSrc Mux:

Input: Reg[x12] and 0x0000000000000014

Output: 0x000000000000014

MemtoReg Mux:

Input: Reg[x13]+0x14 and undefined

Output: undefined

4.5.4

ALU Input: Reg[x13] and 0x0000000000000014

Add Unit 1 Input: PC and 4

Add Unit 2 Input: PC+4 and 0x0000000000000028

4.5.5

The Inputs for the registers unit are:

Read Register 1: 0x13

Read Register 2: 0x12

Write Register: 0x0 or don’t-care

Read Register: don’t-care

RegWrite: false