CS-740 Homework #2 Sample Answers

1. Handling Data Hazards

There are several possible read after write (RAW) data hazards, but no write after write (WAW) or write after read (WAR) data hazards. The possible RAW data hazards are:

(1). RAW without memory reference (pure register operations): In this case, the content of a register consumed by ALU is generated by one or two intermediate previous instructions which will write the results to this register, therefore, we have to add the forwarding circuits which feedback the result from MEM or WB stages. If the content of this register is generated by the previous two instructions, then the result must come from the feedback of MEM stage.

(2). Read After Load: In this case, the content of a register consumed by ALU is generated by one or two intermediate previous LOAD instructions which loads the content of some memory locations into this register. There are two cases. First, if a read immediately follows a load, we must stall the read instruction in the EX stage for one clock and then the needed value will be feedback from the WB stage, because the loaded value will not be available until the MEM stage of the load instruction and it is needed in the EX stage of the read instruction. Seconds, if a read follows a load with one intermediate instruction, the forward circuits from WB to EX will handle this hazard and no stalls are needed.

(3). Store After Load: In this case that a store immediately follows a load, the result of load is needed at the MEM stage of the store instruction, so we have to add another forwarding circuit which feeds back the result from WB stage to the MEM stage and then the store instruction can be performed without installs. It will cost you 10 points, if you didn't handle it.

There are some minor data hazards you should handle. First, you shouldn't forward the values written to register 0 which is always zero. If you didn't check for this, it will cost you 5 points. Second, you should handle the following case:Addiu $2,$1,$1 followed by LW $3,4($1) followed by Addiu $4,$2,$3. The LW instruction will stall the third instruction (Addiu $4,$2,$3) in EX stage for one clock. After one clock, the forwarding value from the first instruction(Addiu $2,$1,$1) will be lost and the value of $2 is not determined in EX stage. You should modified your program to keep the forwarded values. If you didn't handle this, it will cost you another 5 points.

2. Jump Instructions

The destination of Jump instructions is unknown before EX stage. The correct target address of these instructions will come out from EX stage, but at this time three instructions has been fetched into the pipeline. One of these three instructions (Delay slot) is still good, therefore, two bubbles must be issued to cancel the other two instructions. JALR or JAL should also modify the content of register 31 to the address after the delay slot. For the J or JAL, the target address is calculated in EX stage by ((PC+4)&0xF0000000) | the offset left shifted by 2. Some people implement this in the IF stage and then after fetching the delay slot instruction it will jump to fetch the target

instruction. Therefore, no cancelation is needed. This is nice idea, but it changed the structure of the pMIPS pipeline organization and costs you 5 points. For JR or JALR, the target address is the content of the register.

The JR or JALR instructions may cause the RAW data hazard by the register, since these two instruction may use a register whose correct value is modified by its intermediate previous one or two instructions. This hazard can be handle the forwarding circuits mentioned in section 1. The JAL or JALR instructions may also cause the RAW data hazard by the register 31. Similarly, it can be handled by forwarding. If you didn't mention these data hazards, it will cost you 5 points.

3. Control hazards and Structural Hazards

We have to handle the control hazards caused by the branch or jump instructions. For the jump instructions, the branch is always taken and two bubbles must be issued to cancel two instructions already in the pipeline.

For the branch instruction, the assumption is that the branch is not taken. If the branch is taken, then two bubbles must be issued to cancel two instructions already in the pipeline.

The pMIPS has no unites shared between stages and it only issues one instructions per clock, therefore there is no structural hazards.

4. Implementation

Please see the asst2/solve.c.