ECE352F: Computer Organization

Problems 2

Fall 1999

These problems are in addition to those given out for the midterm. The exam will cover material from the entire course.

Chapt. 6 37

Chapt. 7 1, 6, 10, 12, 13, 14

Chapt. 8 2

Here are some additional pipelining questions.

1. We have discussed a 4-stage pipeline with the following stages:

   Fetch fetch instruction

   Decode decode instruction, including accessing registers

   Execute execute the instruction

   Write write results

   Since instruction fetch is supposed to happen in one cycle, what does that imply about the memory system?

   We said that memory accesses were of the form that did the load of a register from memory or a store of a register to memory. They would look like:

   ```
   ld x(r1),r2        ; x is an offset added to contents of r1
                      ; r2 is loaded from memory
   st r1,x(r2)        ; r1 is stored in memory
   ```

   To compute the address, the ALU in the execution unit is to be used to add the offset to the register.

   When is the address available?

   Assume that it takes one cycle to access memory, what needs to be done for memory instructions if you want to maintain the 4-stage pipeline?

If you want to keep the pipeline busy as much as possible what modification needs to be done to the pipeline? You should make sure the write stage for all instructions happens at the same step. Why? Show your new pipeline and explain in detail how the memory instructions get executed.

What must be true about the memory system to allow memory accesses to happen in one cycle?

Does this new pipeline affect the number of branch delay slots?

2. Now consider that you have the following pipeline:

   Fetch

   Decode

   Execute

   Memory

   Write

   This is a modification of the previous pipeline to allow memory accesses to fit into the pipeline. This is what you should have gotten in the previous question.

   Assume that you have ld/st instructions as described above. Branches are of the compare-and-branch flavour:

   ```
   bgt r1,r2,label
   ```

   which means that if r1-r2 > 0, the program branches to label. In the machine code, the branches are relative, so the actual instruction encodes an offset represented in 2's complement notation.

   Show how a branch progresses through the pipeline. In particular, show what is needed to update the PC with the correct value depending on whether the branch is taken or not. Show what values are used to compute the new PC values for a taken branch.

   What datapath resources will you need to make this work? For example, you will need an ALU unit to do the execute unit functions. You'll need some pipeline registers. Don't forget that you'll need to update or modify the PC at times. Draw a diagram of your datapath.

3. We say that an instruction completes when it changes the state of the processor. Why is it important to know when completion occurs? Why is it good to have all instructions complete in the same stage of the pipeline? Hint: consider what needs to happen during an interrupt/exception.

   When does a branch instruction complete in the pipeline of question 2?