330 Project – Spring 2017

1. In this project you will build a simulator that simulates an advanced version of the pipelined MIPS CPU discussed in class, namely R4000 MIPS CPU. You may use C++ / C# or Java to build your simulator. This is a group project with 4 students per group. Every one in the group MUST contribute to the project and write original code.
2. The CPU is required to use dynamic branch prediction using a BTB as discussed in class - using 2-bit predictor, of the pipelining Unit.
3. The main differences between R3000 & R4000 CPU are 8 stages vs. 5 stages. In R4000, IF stage is now split into 2 stages IF1 & IF2. Data memory stage is now 3 stages instead of 1, DF1, DF2, TC., with data available only after 2nd stage. TC - tag check stage is used to make a final tag check in cache. You can assume all cache accesses are cache hits. This means your pipeline will now have 8 stages.
4. MIPS suffers from rich procedure calls; You are required to implement 2 new instructions a) Jump\_procedure & b) return\_procedure; Jump procedure will implement pseudo direct jumping - similar to JAL, but instead of saving in LR, it will save in a 4entry stack you are implementing; Return procedure will return to next instruction by popping last saved entry on stack; Needless to say you need to implement a 4 level stack to allow 4-deep procedure nesting; Use the same assembly code format discussed in the chapter, but simplified. eg you can use numbers 1,2,3 instead of real register names. Register file implementation can be 16 i.e.registers 0-15, same with Memory, 16-32 locations max.
5. Optional Advanced change - Implement a 2-way superscalar version of R4000, which means you have 2 execute pipelines running in parallel. Two instructions can be executed at same time. This is more complex and advanced architecture change, Please see me if you plan to implement.
6. There are three main parts to your project:
   1. Build a translator that accepts MIPS assembly code from an input text file, and produces a parsed format ready for fetch and execution by your simulator. This is equivalent to loading instruction memory with your sample code. You need to perform simple syntax checks on your input text file to ensure syntax correctness. Your CPU must accept the following MIPS instructions[ ADD,ADDI,XOR,LW,SW, BLE, J, SLT, JAL, JR].
   2. Build the simulator itself that accepts the above parsed code and runs your code (simulates your CPU). Your simulator must reproduce on a clock by clock basis the pipelined execution environment, and produce and print the results.
   3. You must be able to display on the screen every clock shown as a pipeline timing diagram Please see sample Project document from Mai & Merihan:
      1. Contents of each pipeline stage, e.g. IF contains instruction 4 / Add,
      2. contents of register file,
      3. contents of Memory
      4. contents of the branch target buffer
      5. contents of the procedure stack
7. You will need to produce a pipeline flow diagram [ X axis is time / clock cycle, Y axis is instructions executed] You will also need to print out total clocks used in any simulated run. NOTE: each stage in your pipeline must perform the tasks that are normally done in that stage. Do not move the task to the translator or any other stage.
8. Test your simulator by running a brief program (6 instructions) with enough instruction mix to demonstrate your simulator.
9. Each group will meet with me last week of semester, by appointment, to demonstrate their project and answer questions. Your final report must include Documented code [ with comments ], screen shots of sample runs showing showing pipeline timing diagrams

You must be able to run on a clock by clock basis, showing the contents of all relevant registers and memory locations [keep it small] Students are free to discuss project ideas and implementations, but cannot share code.