**Project #2**

1. Project Objective:
   1. Implement a pipelined, functional processor simulator for the reduced MIPS R3000 ISA, following the specification “*Datasheet for the Reduced MIPS R3000 ISA*” in *Appendix A.*
   2. Design a test case to validate your implementation, particularly on hazards handling.
2. Specification:
   1. Pipeline architecture
      1. 5 stages: instruction fetch (IF), instruction decode (ID), ALU execution (EX), data memory access (DM) and write back (WB).
      2. To avoid re-execution of some stages, the order of simulating the pipeline is WB -> DM -> EX -> ID -> IF.
      3. **For load/store instructions, compute the address to be accessed in the D memory during the EX stage, and access data memory during the DM stage.**
      4. **Do branch prediction in the IF stage and verify the correctness in the ID stage.**
      5. Arithmetic/bitwise shift/logical operations are done during the EX stage.
      6. There are three forwarding paths: EX/DM to ID, EX/DM to EX, DM/WB to EX.
      7. All **write back** targeting to registers $0~$31 **are executed during the first half of the cycle** in the WB stage.
      8. All **reads from registers are done during the second half of the cycle** in the ID stage.
      9. PC and branch history table ( BHT ) are updated in each cycle **after** the execution of all instructions in the pipeline.
      10. If inserting “NOPs” is needed to resolve hazards, use *sll $0, $0, 0*, i.e. the bit stream 0x00000000. **In your implementation, you should decode** the instruction bit stream 0x00000000 **as NOP**.
      11. **Conditional branches and unconditional branches are evaluated during the ID stage.**
      12. **The instruction following unconditional branches always be flush, that is, there is always an NOP following unconditional branchs.**
   2. Branch prediction
      1. Construct a 2-bit branch history table (BHT) for branch prediction and a branch unit for checking whether the instruction is a branch instruction and calculating the target address.
      2. Prediction happens in the IF stage by utilizing the PC[2:5] bits (4 bits) of the PC address as an index to look up BHT.
      3. Note that we need to verify the correctness of prediction in the ID stage by evaluating the branch condition.
      4. All entries in BHT are initialized as bits “01”, i.e. non-taken .
      5. The BHT is updated only after the branch condition can be evaluated correctly, that is, there should have no data hazard when you update the BHT ."
      6. For more details please refer to *Appendix D.*
   3. Other Constraints
      1. The pipeline is **initialized with NOPs in all stages**.
      2. The simulation of the pipelined processor **terminates after the first “halt” instruction arrives at the stage WB**.
      3. **Register $0 is hard-wired to be 0**; any attempt to write to register $0 takes no effect.
      4. The instruction memory is of 1K size, the data memory is also of 1K size.
      5. The executable should be named **pipeline**.
3. Input Format:

It’s the same as that of Project 1. Please refer to the specification of Project 1 and *Appendix B*, “*Sample Input*.”

1. Output Requirement:

Generate an output file named **snapshot.rpt** for each case tested. The file should contain the values of registers, mnemonic in each pipeline stage, hazards and prediction information at each cycle. For format details please refer to *Appendix C-2*, “*Sample Output for Project 2*.”

1. Test case design:

Your test case should cover at least one control hazard, one data hazard resolved by inserting NOPs, and one data hazard cleared by forwarding. The number of NOPs inserted is not limited, The remain is same as project 1.

**Note that your test case should not execute more than 30,000 cycles and should not cause initialization error or it will be deemed invalid.**

**Any error which cause simulator to halt (defined in appendix of proj1 ) should not be in**

**your test case. (No address overflow error、No misalignment error)**

**The definition of initialization is the action that load your iimage.bin /dimage.bin files into your I-Memory/D-memory.**

1. Evaluation:

Same as project 1.

Note that all evaluations, verifications will be done on **nthucad workstation (on backend ic17 )**. Furthermore, we will evaluate your project using scripts. Please make sure that your project can be executed by the script provided by TA’s. **If it cannot run through the script, you will lose all your points even if your program or result is correct.**

1. **Etiquette**
   1. **Do not copy others’ works, or you will fail this course.**
   2. **No acceptance of late homework.**
   3. **For details of submission, please note the announcement on the course website.**