Section 1: Title page

. Project title

Team name: Team T.L.D.C.S. (Team Learned Designers Collaborating Sensibly)

Members:

Shuying Fan – Design Master

Donald Pomeroy – Verification Master

Corey Stappenbeck – Integration Master

Leonard Robinson - Verification Master

Tong Zhang – Design Master

. Project member duties

o Elect a Design Master

o Elect a Verification Master

Section 2: Design Overview (A simple plain English description)

Basic techniques to exploit instruction level parallelism such as loop unrolling, branch prediction, dynamic scheduling, and speculation only suffice to reduce the Clocks Per Instruction (CPI) to a minimum of 1. If we seek to improve performance further, we must reduce the CPI to less than 1.Multiple-issue processors help us achieve this goal by permitting multiple instructions to issue in a

clock cycle.

Your design should allow for the following:

1. Allow fetch, decode, issue and commit of two instructions every cycle as possible.

2. You will need to create a multi--‐ported register file.

3. Assume four ALUs and multiple copies of other processing elements as needed.

4. Allow out--‐of--‐order execution.

The pipeline will attempt to fetch, decode, issue and commit 2 instructions each cycle.

The ideal achievable CPI is, therefore, 2.

Note that you will be required to implement Register renaming

(cf. H&P, more materials available on request) and speculative execution with simple always‐taken branch prediction.

A register file includes 32 registers are included in this pipeline design.The strategy of branch prediction is always taken.The superscalar out-of-order MIPS pipeline can fetch two instructions every cycle from the memory.

To avoid RAW hazard, before fetching a RAW\_checker is needed. If there is no RAW hazard, the pipeline will fetch two instructions every cycle; if not, a certain number of stalls will be inserted before checking.

After fetching stage, the decoder will decode the instructions. Register renaming is implemented to deal with RAW and WAR hazards after decoding. A look-up table which records the renaming information is created in this stage.

Then the instructions go into the issue queue. Since all the data dependences have been dealt with in the previous stages already, the issue logic only needs to issue two instructions every cycle whenever there are available execution units. The issue queue is also able to flush the queue when there is branch mis-prediction.

After execution, the instructions can be committed and write back to the memory.

Section 3: Unit Level interfaces

Section 4: Subunit partitioning and interfaces, Test harness structure

Section 5: Microarchitecture design

Section 6: Verification strategy

Section 7: Performance estimates

Section 8: Area estimates

Section 9: Bugs, Coverage,

Section 10: Document revision history