Homework 1
MIPS R10000

This exercise is based on and refers to the *MIPS R10000 Superscalar Microprocessor* document, which can be found on the course web page.

1 Pipeline and Register Renaming

1. Explain why in the 3rd stage of the pipeline (see Figure 2) the *register file* is read in the 2nd half of each cycle. What are the advantages and disadvantages of such an implementation?

2. Figure 5 describes register renaming. Why are the destination and source registers represented with 5 bits in the original instruction, and with 6 bits after the renaming?

3. Specify the function of the *queues*, *active lists*, and *map tables* in the R10000. Relate each of these to the structures described in the course (*reservation stations*, *ROB*, ...).

4. Describe the role of the components *Rdy*, *OpA*, *OpB*, *OpC*, *Dest*, *Old Dest*, *Log Dest*, *Tag*, and *D* in Figure 5.

5. How many memory bits are needed for the *FP queue*, the *active list* (without the *CC* bits), the *FP register map table*, the *free register list*, and the *FP busy bit table*? Justify your answers.

6. How many read ports are there in the *FP busy bit table*? How many read ports and how many write ports are there in the *FP map table*? Justify your answers.

7. Explain how each kind of dependency is detected and how each is handled.

8. Using the tables in annex, describe the behaviour of the processor executing the following program[1]. You should complete the *active lists*, the *FP queue* and *address queue*, the *FP register map table*, the *busy bit table*, the *free list*, and the position of the instructions in the pipeline for each cycle. Use one copy of the tables per cycle (in Excel, use a separate sheet for each cycle). Write all numbers in hexadecimal. Use the following hints and assumptions to simplify the simulation:

    - Assume the data and instruction caches never raise a miss.
    - Assume all queues are empty at the beginning.
    - Assume integer register 1 ($I1) is mapped to physical register ($I0x01).
    - Assume all instructions after #00000010 are NOPs.
    - Assume that after issue, all instructions spend exactly one cycle in each applicable stage—even, e.g., the SQRT.S instruction.
    - Assume floating point results are first available in the write-back stage.
    - Remember that internal components are usually structured as they are for a good reason (or reasons).
    - Remember that creating new register mappings, removing newly-allocated registers from the free list, and marking those registers as busy must all occur in the same cycle.
    - Remember to carefully consider the timing and sequence of register write-back, instruction issue, removal from instruction queues, and active list updates.

```
#00000001    LDC1     $F0, #0000($I1)
#00000002    LDC1     $F1, #0001($I1)
#00000003    LDC1     $F2, #0002($I1)
#00000004    LDC1     $F3, #0003($I1)
#00000005    MUL.S    $F4, $F0, $F3
#00000006    MUL.S    $F5, $F1, $F2
#00000007    SUB.S    $F4,$F4, $F5
#00000008    ADD.S    $F5, $F0, $F1
#00000009    MUL.S    $F9, $F5, $F5
#0000000A    MUL.S    $F10, $F4, $F0
#0000000B    SUB.S    $F9, $F9,$F10
#0000000C    SQRT.S   $F9, $F9
#0000000D    SDC1     $F4, #0004($I1)
#0000000E    SDC1     $F5, #0005($I1)
#0000000F    SDC1     $F9,#0006($I1)
#00000010    SDC1     $F10, #0007($I1)
```

[1]LDC1 = load double word from memory
SDC1 = store double word to memory
instr.S = floating point instruction (example multiplication = MUL.S)

2 Exception Handling

1. When an exception occurs, the processor must restore its state to exactly match the programmer's expectation precisely when the programmer expects that exception to arise (for example, all the instructions executed after the faulty instruction should be discarded). Explain how the R10000 achieves this goal.

2. In the previous program, what happens if instruction #00000006 generates an overflow exception? Describe how the processor restores its state according to information it has in the queues and active lists and why the state is restored in this way.

3 Branch Prediction

1. The R10000 is able to predict branches. What kind of algorithm does it use? Is it dynamic or static prediction? Describe the state machine the algorithm uses.

2. What happens if the prediction is not correct? Describe the processor's behaviour in detail.

3. Explain why there is a NOP after the branch in the following program. Change the program to avoid such an instruction.

```
       #00000001    LD     $I1, #0000($I10)
       #00000002    SUB    $I17,$I17, $I17
loop:  #00000003    LD     $I5, #0000($I2)
       #00000004    LD     $I6, #0000($I3)
       #00000005    MUL    $I5, $I6, $I5
       #00000006    SD     $I5, #0000($I4)
       #00000007    SUBI   $I1, $I1, 1
       #00000008    ADDI   $I2, $I2, 4
       #00000009    ADDI   $I3, $I3, 4
       #0000000A    ADDI   $I4, $I4, 4
       #0000000B    BEQ    $I1, $I17, loop
       #0000000C    NOP
       #0000000D    LD     $I1, #0000($I10)
       #0000000E    MULI   $I1, $I1, 4
       #0000000F    SUB    $I4, $I4, $I1
```