EE 4720 **Homework 4** Solution **Due: 28 November 2001**

Problem 1: Solve Problems 3 and 4 from Fall 2000 Homework 5, available via `http://www.ece.lsu.edu/ee4720/2000f/hw05.pdf`. Using the solutions at `http://www.ece.lsu.edu/ee4720/2000f/hw05_sol.pdf` assign yourself a grade in the range $[0, 1]$. Either: indicate the grade you assigned yourself or write "Did not solve." A solution can be provided along with a grade. It will be corrected but your grade will be used. If you opt not to solve it you will receive full credit but will be hurting your ability to solve future problems.

For the following questions read Kenneth C. Yeager, "The Mips R10000 Superscalar Microprocessr," IEEE Micro, April 1996, pp. 28-40. A restricted-access copy can be found at `http://www.ece.lsu.edu/ee4720/s/yeager96.pdf`. *Access is allowed from within the* `lsu.edu` *domain or by using the userid "ee4720" and the correct password. Though not needed for this assignment, information on the MIPS64 4 ISA (implemented by R10000) can be found in* `http://www.ece.lsu.edu/ee4720/mips64v1.pdf` *and* `http://www.ece.lsu.edu/ee4720/mips64v2.pdf`.

Skip over the material on the memory system (under heading "Memory Hierarchy") and the system interface. Material related to memory will be covered later in the semester.

Problem 2: The paper uses the four terms below, for each show the corresponding, or most similar, term used in class.

- Graduate⟶Commit
- Active List⟶Reorder Buffer
- Tag⟶Reorder Buffer Entry Number
- Logical Register⟶Architecturally visible register number.

Problem 3: For the superscalar processors described in class taken branches resulted in higher than ideal CPI; the higher the fetch/decode width (the n in n-way superscalar) the worse the problem was. Why is this problem not as severe in the R10000? (Branch prediction is not the answer.)

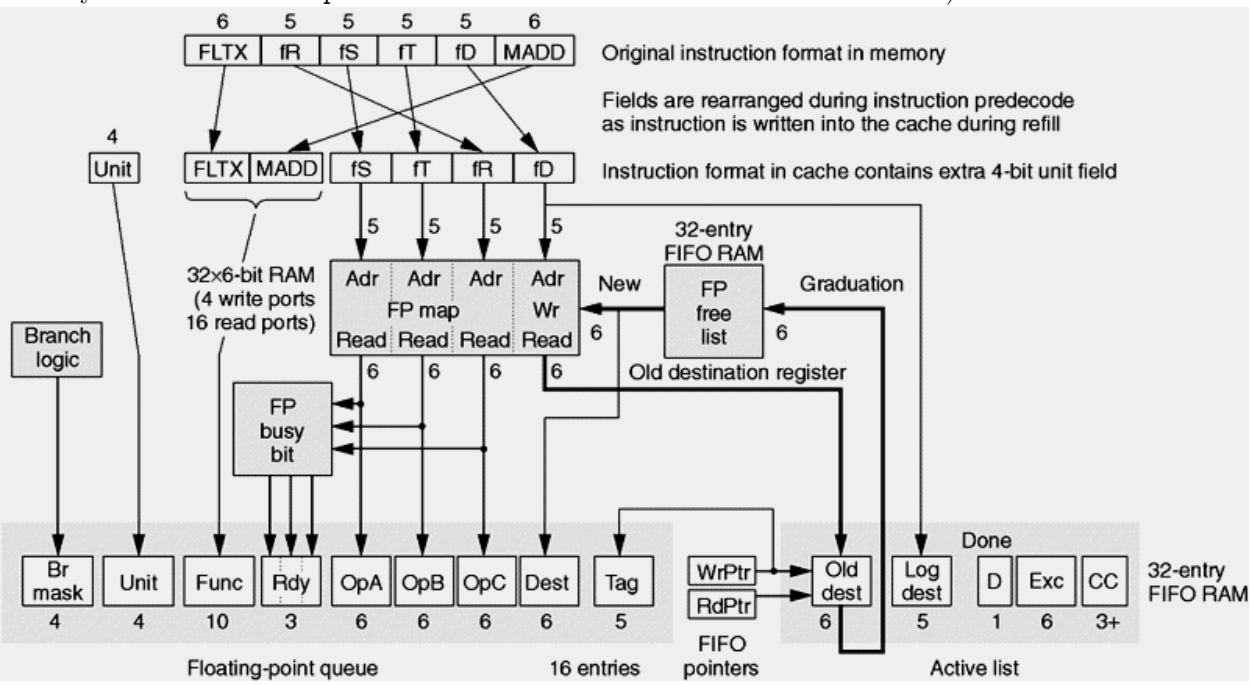
Because a group of fetched instructions does not have to be 16-byte aligned (that is, the address of the first instructions need not be a multiple of 16).

Problem 4: The MIPS R10000 does not have anything like a commit register map or a commit free list. (The register map and free list at the bottom of the figure from the class notes on the next page.) How were those used with exceptions in the Method-3 dynamically scheduled processor described in class? How does the R10000 deal with exceptions given their absence? Do not describe the entire exception process, just those pieces of hardware and steps needed to do what was done with the commit free list and register map.

In the implementation described in class, when the faulting instruction reaches the head of the reorder buffer the reorder buffer is flushed and the commit register map and free list are copied to the ID register map and free list, respectively.

In the R10000 the active list holds previous physical register assigned to the destination. When the faulting instruction reaches the head of the active list, rather than flushing the active list, the hardware uses the elements in the active list to repair the register map. For each active list element starting with the one holding the faulting instruction, the register map element corresponding to the destination register is written with the previous physical register. This processes reverses the changes to the register map made by the faulting instructions and those that followed it.

Problem 5: Figure 5, reproduced below, shows the information that will be placed in the active list and floating-point queue for an instruction being decoded. Various field names are shown along the bottom of the figure. The instruction format fields are shown at the top. Fields `fR`, `fS`, `fT` are source registers (not every instruction uses three); field `fD` is the destination register, `FLTX` is the opcode, and `MADD` is an extension of the of the opcode field (as `func` is in DLX). (The figure appears to be using field values rather than names for the first and last fields. `MADD` is the name of an instruction, multiply-add, and `FLTX` may be an abbreviation for floating-point extended, though the architecture manual calls the field value `COP1X`. They probably should have used `opcode` instead of `FLTX` and `function` instead of `MADD`.)



Write the field names from the bottom of Figure 5 next to the corresponding fields in the figure, from the class notes, below. Though Figure 5 shows a floating-point instruction assume that integer instructions are handled the same way. Some fields have no analog in the figure below, these can be omitted; `Tag` is **not** a field that can be omitted.

Field names shown in blue

.