CpE 313 Fall 2004
Microprocessor Systems Design
Exam 2

November 9, 2004

Instructions

Read each individual problem appearing on this exam carefully and do only what is specifically stated.

This exam is **70 minutes** long. You are not allowed any reference materials other than two sides of a letter sized page. Calculators are allowed but you are not allowed to use the calculator memory to store reference material related to this course. On your initial pass through the exam, skip any problems that appear to be overly difficult.

The exam consists of 20 multiple choice problems. **You only have to solve 17 problems correctly to get full credit.** Any additional correctly solved problems will earn you extra credit that you can use elsewhere in the course. No partial credit will be given for a problem. **Please circle your choices clearly.**

IMPORTANT: Put down your initials at the TOP of EACH page. Also, be sure to read and sign the Academic Honesty Statement that follows:
"In signing this statement, I hereby certify that the work on this exam is my own and that I have not copied the work of any other student while completing this exam. I understand that, if I fail to honor this agreement, I will receive a score of ZERO for this exam and will be subject to possible disciplinary action."

Printed Name: ..... Signature: .....

DO NOT BEGIN UNTIL INSTRUCTED TO DO SO.

Problem 1

For a unified cache, why does a data load or data store hit take more time than an instruction load hit?

(a) The unified cache has a lower miss rate than two split caches of the same combined capacity.

(b) The unified cache has a higher miss rate than two split caches of the same combined capacity.

(c) There is only one set of cache ports for instruction and data memory references, and data memory references access it first.

(d) There is only one set of cache ports for instruction and data memory references, and instruction memory references access it first.

Problem 2

In general, increasing the set associativity of a cache reduces its AMAT value. However, the preceding statement is not likely to be true for a

(a) small cache with a low (1-2 way) associativity.

(b) large cache with a low (1-2 way) associativity.

(c) small cache with a high (8 way) associativity.

(d) large cache with a high (8 way) associativity.

Problem 3

What is the CPI for a computer when the miss penalty is 40 cycles, all instructions take 5 cycles to execute (unless stalled), miss rate is 7%, and there is an average of 33% data load/store instructions? Assume that any stalls in the program are only from instruction or data memory accesses.

(a) 8.72.

(b) 3.72.

(c) 5.92.

(d) 7.80.

Problem 4

Assume we have a 2-way set associative cache with a total of sixteen frames and a First-In-First-Out replacement policy. The cache, **initially empty**, receives the following sequence of block addresses:

10010
00010
10110
10010

Will the fourth memory reference be a hit or a miss, and in which cache frame will the requested block be placed?

(a) hit, and placed in either frame 0100 or frame 0101 (i.e., set 010).

(b) hit, and placed in frame 1001.

(c) miss, and placed in either frame 0100 or frame 0101 (i.e., set 010).

(d) miss, and placed in frame 1001.

Problem 5

Consider the following statements, and then pick the choice, one of (a) through (d), that is most appropriate. How does a direct mapped cache determine if it contains a requested word?

1. It checks to see if the valid bit is asserted.
2. It compares the tag of the requested memory address to the tag stored in the cache to see if they are equal.
3. It compares the block offset of the requested memory address to the block offset stored in the cache to see if they are equal.
4. It compares the index of the requested memory address to the index stored in the cache to see if they are equal.

(a) 1 only.

(b) 1 and 2 only.

(c) 1, 2, and 3 only.

(d) 1, 2, 3, and 4.

Problem 6

Consider a MIPS **integer** pipeline that must maintain precise exceptions. How does servicing an instruction's interrupts in its WB stage ensure that all interrupts in the program will be serviced in the un-pipelined instruction sequence?

(a) When instruction i is in its WB stage, it is certain that instruction $i+1$ is in its MEM stage.

(b) When instruction i is in its WB stage, it is certain that the previous instructions are finished and that their interrupts (if any) have already been serviced.

(c) If instruction i is in its WB stage, it cannot raise an interrupt for that stage because MIPS has no interrupts that can arise **within** the WB stage.

(d) Unlike other stages, WB stages are not executed simultaneously with other stages.

Problem 7

Consider a MIPS **floating point** pipeline that must maintain precise exceptions. Why cannot all the exceptions be "posted" and then serviced in the WB stage of the pipeline?

(a) Because an exception may happen in some stage other than the WB stage. Servicing exceptions in WB will cause unnecessary delay.

(b) Because precise exception handling requires that all instruction execution must stop once the interrupt is raised.

(c) Because not all instructions go through the WB stage of the pipeline.

(d) Because different instructions have different lengths for the execution stage. If a longer instruction causes an exception, a shorter instruction that comes later in the program order might write back its result before the exception is handled.

Problem 8

Observe the following code, which will be executed on a dynamically scheduled pipeline that uses a scoreboard. Assuming that the "div" instruction is in the execution stage, what will be the last stage completed by the "mult" instruction? Assume that there are enough functional units for all three instructions to execute simultaneously.

```
DIV    F1, F2, F3
ADD    F4, F1, F0
MULT   F4, F9, F11
```

(a) IF.

(b) IS.

(c) RO.

(d) EXE.

Problem 9

Which one of the following is **not** a disadvantage of the scoreboard method as compared to Tomasulo's algorithm?

(a) Scoreboard requires that all operands be read from register file.

(b) Scoreboard requires that all operands be read at the same time.

(c) Scoreboard resolves a WAR hazard by stalling the writing instruction in the WR stage.

(d) Scoreboard cannot resolve WAW data hazards.

Problem 10

Consider a dynamically scheduled pipeline that uses Tomasulo's algorithm and does not use a re-order buffer. What is done if the functional unit needed for an instruction is available, but only 1 of the 2 needed operands is available?

(a) The instruction stalls in the issue stage until both operands are available at the same time.

(b) The instruction stalls in the RPO stage until both operands are available at the same time.

(c) The current values of both operands are read regardless of their validity.

(d) The reservation station stores the available operand and the tag for the functional unit that will produce the pending operand.

Problem 11

Consider a dynamically scheduled pipeline that uses Tomasulo's algorithm and does not use a re-order buffer. How many clock cycles does it take to read the available operands, read the tags of the unavailable operands, and complete register renaming in the register file?

(a) 1.

(b) 2.

(c) 3.

(d) Cannot be determined without knowing the data dependencies of the given instruction.

Problem 12

Consider a dynamically scheduled pipeline that uses Tomasulo's algorithm and does not use a re-order buffer. What type of data hazard is eliminated by register renaming?

(a) WAR.

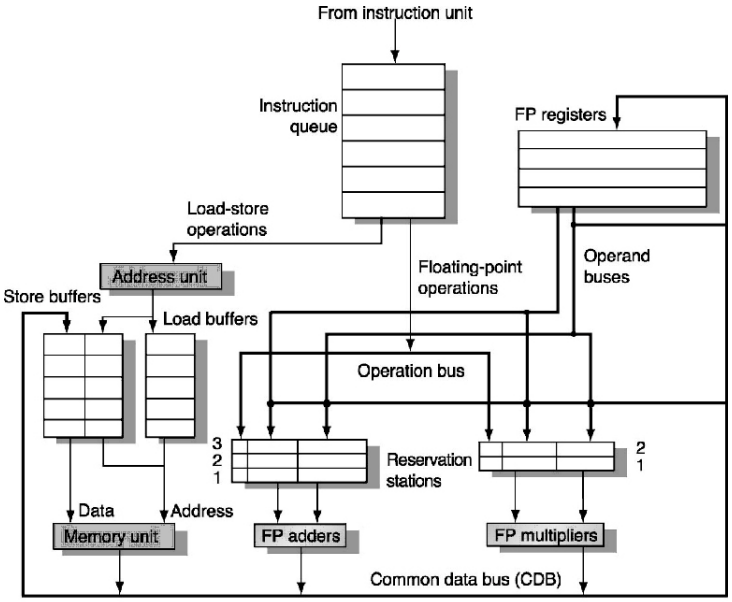
(b) WAW.

(c) RAW.

(d) Both a and b.

Problem 13

With respect to the diagram below, how does a floating point functional unit determine if the instruction in a given attached reservation station is ready to be executed?



(a) It accesses the register file to see if the register file tags for the two source operands are 0. If yes, no operands are pending and the instruction can be executed with the operand values already present in the reservation station.

(b) It accesses the register file to see if the register file tags for the two source operands are 0. If yes, no operands are pending. The FU then transfers the operands from the register file to the reservation station. The instruction can then be executed.

(c) It accesses the Qj and Qk fields within the reservation station. If both Qj and Qk are zero, it means that the instruction is ready to be executed.

(d) It accesses the "busy" field in the reservation station. If it is set to "no," the reservation station is done waiting on operands, and the instruction can be executed.

Problem 14

Consider a dynamically scheduled pipeline that uses Tomasulo's algorithm and does not use a re-order buffer. If the system has 8 floating point registers, F0–F7, with the register tags (i.e., the Q values stored in the register file) listed below, which one of the following statements is **definitely** true?

F0: 0
F1: 2
F2: 7
F3: 3
F4: 5
F5: 0
F6: 6
F7: 0

(a) There are 5 instructions stuck in the RPO stage.

(b) There are 3 instructions waiting on pending operands.

(c) There are 5 instructions that have been issued but have not yet written their results.

(d) There are at least 5 instructions that have been issued but have not yet written their results.

Problem 15

Consider the following code segment to be executed on a dynamically scheduled pipeline that uses Tomasulo's algorithm and does not use a re-order buffer.

```
DIV   F0, F1, F3
SUB   F12, F0, F4
```

Immediately before the IS stage of SUB began, the register file entry for F0 contained Q = 3 and V = 348. At the same time, the register file entry for F4 contained Q = 0 and V = 13. Assume that the DIV instruction takes much longer to execute than the SUB instruction. Immediately after SUB finishes its IS stage, which one of the following can be ***definitely*** said about the Vj, Vk, Qj, and Qk fields in SUB's reservation station?

(a) Vj = 348, Vk = 13, Qj = 0, Qk = 3.

(b) Vj = 348, Vk = 13, Qj = 3, Qk = 0.

(c) Vj = some irrelevant value, Vk = 13, Qj = 3, Qk = 0.

(d) Vj = 13, Vk = some irrelevant value, Qj = 0, Qk = 3.

Problem 16

Assume a MIPS integer pipeline without any forwarding. Which piece of code cannot possibly benefit any more from dynamic scheduling than from static scheduling? Assume that the pipeline design will never change, and therefore one compiler will work for all programs written for this machine.

(a)
```
SUB   R5, R1, R4
ADD   R9, R5, R4
SW    20(R1), R9
LW    R8, 32(R3)
```

(b)
```
LW    R4, 20(R1)
SUB   R5, R1, R4
```

(c)
```
ADD   R5, R6, R4
SUB   R1, R5, R3
ADD   R2, R4, R6
```

(d)
```
LW    R4, 20(R1)
SW    20(R6), R4
```

Problem 17

Two-way set associative caches have better miss rates than direct mapped caches of the same cache size, but at the same time suffer with higher hit times. Which one of the following is an attempt to reduce a two-way set associative cache's hit time?

(a) When a branch is fetched and decoded, predict the direction taken by branch. Execute along the predicted path until the branch instruction comes up at the head of the re-order buffer. Commit the branch if the prediction is proved to be correct. Otherwise flush the re-order buffer, and re-start the execution from the branch target.

(b) While tag match is being performed, pick one of the blocks in the indexed set, use block offset to select a word from it, transfer the word to the CPU, and use it speculatively. If it is later found that the tag match had failed for the picked block, set a "speculation" bit to indicate this. When the speculated load instruction comes up at the head of the re-order buffer, commit it if the speculation bit is 0. Otherwise flush the re-order buffer, and re-start the execution from the load instruction, this time checking both tags before transferring the word.

(c) Increase the size of the cache.

(d) Issue a load instruction without first ensuring that its address does not match addresses from any prior stores. When the load instruction comes up at the head of the re-order buffer, ensure that its address does not match any prior issued stores. If there is no match, commit the load. Otherwise flush the re-order buffer, and re-start the execution in correct SW-LW order.

Problem 18

Consider the following code segment, which is to be executed on a dynamically scheduled pipeline that uses Tomasulo's algorithm and does not use a re-order buffer.

```
ADD    F8, F8, F4
SUB    F8, F8, F2
```

Assume that there are enough functional units for both instructions to be executed simultaneously. If neither instruction has yet completed its WR stage, but both instructions have been issued, which value is currently stored in the register file for the Q tag of F8? Assume further that the RS tag for the functional unit of the ADD instruction is 1, and for the SUB instruction is 2.

(a) 0.

(b) 8.

(c) 1.

(d) 2.

Problem 19

Consider a dynamically scheduled pipeline that uses Tomasulo's algorithm and does not use a re-order buffer. How does such a pipeline handle a potential store-to-load RAW hazard?

(a) If a load instruction is issued and there are store instructions in the store buffer, the load instruction waits in the load buffer until all store operations have been completed.

(b) Before sending a store instruction to the store buffers, the store address is calculated and compared to the addresses in the load buffer. The store instruction waits only if there is an address match.

(c) Before sending a load instruction to the load buffers, the load address is calculated and compared to the addresses in the store buffer. The load instruction waits only if there is an address match.

(d) Tomasulo's algorithm cannot avoid a store-to-load RAW hazard unless a re-order buffer is employed to perform speculative execution.

Problem 20

If the ready field of a particular re-order buffer entry is set to "yes," then

(a) the corresponding instruction's results have been written to the register file or to memory.

(b) the corresponding instruction has completed the WR stage but not yet committed.

(c) the corresponding instruction has been issued but not yet completed the WR stage.

(d) the corresponding instruction is ready to be executed.