CPSC 313, 04w Term 2— Midterm Exam 2

Date: March 11, 2005; Instructor: Mike Feeley

This is a closed book exam; no calculators; no notes. Answer in the space provided; use the backs of pages if needed. There are **8** questions on **6** pages, totaling **50** marks. You have **1 hour** to complete the exam.

NAME: ____ **SCORE:** _____ / 50

STUDENT NUMBER: ____________________

1. **(10 marks)** Short answers.

1a. Give an example of one important CISC feature that is normally not part of a RISC instruction set architecture. Very briefly explain why by referring back to the y86 pipeline implementation.

1b. Write a sequence of two C statements that are causally dependent on each other and show what causes the dependency.

1c. Define temporal locality and say why it important for caching?

1d. Define instruction-level parallelism. Is it important for pipelining or caching? Why?

1e. Give one advantage and one disadvantage of a direct-mapped cache compared to an associative cache (i.e., fully-associative or set-associative).

_____ / 10

2. **(6 marks)** Pipelines.

2a. Carefully explain why one would expect that changing a processor implementation from sequential to pipelined would improve the processor's performance.

2b. List all of the reasons you can think of why adding more stages to a pipelined implementation might not necessarily improve performance.

2c. Does it make sense for Intel to report a single throughput measurement for the P4 Prescott? Why or why not?

3. **(3 marks)** Object files and linking.

3a. What is the benefit of separate compilation and what problem does it cause that linking solves?

3b. What is the difference between a *relocatable* and an *executable* object file? How do they relate to the first part of this question?

_____ / 9

4. **(8 marks)** You are responsible for designing the next-generation y86 cpu. The design challenge you are confronted with is that the execute phase has roughly twice the delay as any of the other pipeline phases. You decided to split this stage in two, so that the pipeline now has six stages and none of the results of the execute phase are available until the end of the second execute stage. Answer the following questions.

4a. What problem did the slow execute phase cause and how did splitting it in two solve this problem?

4b. Recall that a data hazard exists in the following scenario (among others):

```
addl    %eax, %ebx
rrmovl  %ebx, %ecx
```

Briefly explain how the current, 5-stage, y86 implementation handles this particular hazard and state whether doing so requires any pipeline stalls or bubbles.

4c. Your 6-stage pipeline can not handle the hazard illustrated by this example in the same way as the 5-stage pipeline. Carefully explain why not. Are additional pipeline stalls or bubbles needed to correctly handle the two-instruction sequence above (assuming no other change to the design)? How many?

4d. Can you modify the design to improve the situation for the two-instruction sequence above? Give a brief, high-level outline of your solution (implementation details are not necessary).

4e. Can you use the same technique for the following two-instruction sequence?

```
addl    %eax, %ebx
addl    %ebx, %ecx
```

Carefully explain why or why not.

_____ / 8

5. **(7 marks)** Consider the following y86 program and the standard y86 pipeline implementation (i.e., PIPE) described in class and in the textbook.

```
start:      rrmovl  %eax, %ebx
            rmmovl  %ebx, 8(%ebp)
            mrmovl  12(%ebp), %ebx
            addl    %ebx, %ecx
            call    foo
            addl    %eax, %ecx
            jne     notzero
            irmovl  $1, %ecx
notzero:    rmmovl  %ecx, 12(%ebp)
            ...
            hlt
foo:        irmovl  $100, %eax
            ret
```

5a. Annotate this code to indicate every place that a pipeline stall occurs and explain why it occurs in each case.

5b. Modify this piece of code to eliminate as many pipeline stalls as you can, but without changing its semantics (i.e., its meaning). You can view this code in complete isolation from any other code; that is, you can assume that no other code calls into this code. Write your modified code below, **using comments to highlight and explain every change you have made**.

6. **(4 marks)** When caches help.

6a. What feature of a cache is designed to exploit temporal locality?

6b. What feature of a cache is designed to exploit spatial locality?

6c. Can a cache provide any benefit to a program that has neither temporal nor spatial locality? How?

7. **(6 marks)** You work for Intel's cache design group and are having a bad day. You've just finished the design of the cache you are responsible for. Now your boss is pushing you to make one more change and you don't want to. For each of the following suggested changes give a good reason why the change might make things worse, assuming that the total size of the cache (i.e., the total amount of data it can store) must remain the same.

7a. Increase the block size.

7b. Decrease the block size.

7c. Decrease the set associativity (e.g., from 8-way to 2-way).

7d. Increase the set associativity (e.g., from 8-way to 32-way).

7e. Changing from write-through to write-back.

_____ / 6

8. **(6 marks)** Consider a small 64-B, 2-way set associative, write through, write allocate cache with 8-B blocks and using LRU replacement. The highest-order address bits are assigned to the tag.

8a. If addresses are 32 bits long, how many address bits are needed for each of the following:

- block offset (b) =
- set index (s) =
- tag (t) =

8b. The cache is initially empty and then it sees the following sequence of accesses. For each access, indicate whether it is a hit or miss. If its a miss, give the type of miss.

- read 0x10:
- read 0x04:
- read 0x14:
- write 0x08:
- read 0x0c:
- read 0x24:
- read 0x10: