UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER SCIENCE THREE

COMPUTER ARCHITECTURE

Tuesday 10 September 2002

09:30 to 11:00

Examiners:

Prof. K. Turner (External)

Prof. C. Stirling (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. (a) Discuss the differences between the following ISA (Instruction Set Architecture) approaches: stack, accumulator, register-memory, and load-store. [*5 marks*]

   (b) Discuss how pipelining can improve the processor's performance. [*5 marks*]

   (c) Explain control hazards, structural hazards, and the 3 types of data hazards, in the context of pipelined processors. Discuss mechanisms to reduce or eliminate the impact of each of these types of hazards. [*10 marks*]

   (d) Explain how Tomasulo's algorithm is a more powerful mechanism for dynamic instruction scheduling than the Scoreboard approach. [*5 marks*]

2. As a computer architect at ED Computers your job is to design the memory hierarchy for the company's new processor. This processor is very fast and careful design of the memory hierarchy is necessary to achieve a low average memory access time. Additionally, ED Computers is considering to use a new chip manufacturing technique that allows the integration of main memory (DRAM) together with the processor inside the chip. After discussing technological constraints with the manufacturing team, you are given the following options for the memory hierarchy:

   (i) on-chip L1 of size 16KB and hit access time of 1 processor cycle, plus, on-chip L2 of size 64KB and hit access time of 10 processor cycles, plus off-chip main memory with access time of 100 processor cycles

   (ii) on-chip L1 of size 128KB and hit access time of 2 processor cycles, plus, off-chip L2 of size 1MB and hit access time of 20 processor cycles, plus, off-chip main memory with access time of 100 processor cycles

   (iii) no on-chip cache, plus, on-chip main memory with access time of 20 processor cycles

   (iv) on-chip L1 of size 16KB and hit access time of 1 processor cycle, plus, on-chip main memory with access time of 30 processor cycles

   After some experimentation with some programs, you have the following figures for cache miss rates for the different cache configurations:

   - 16KB L1 plus 64KB L2: 0.10 miss rate for L1, 0.2 miss rate for L2
   - 128KB L1 plus 1MB L2: 0.08 miss rate from L1, 0.15 miss rate for L2

   (a) Compute the average memory access time for each of the configurations. Which configuration has the best performance? *[10 marks]*

   (b) Assume that the frequency of memory operations in the workload is 10%. Consider that the base CPI of the processor with a perfect memory is 1. What is the average CPI for the processor with each memory system? *[5 marks]*

   (c) Now you are asked to consider how the system would perform when it is used to perform a vector scaling operation ($x = x*y+c$, where: x and y are vectors and c is a scalar). When running at full speed (no memory stall cycles) the processor can compute one element of x in 2 cycles. This means that the processor will perform 1 read of x, 1 read of y, and 1 write of x in every 2 cycles. Ignore all other memory references. Assume that the vectors have double precision floating point elements (8 bytes) and that cache blocks are always 64 bytes long. Would your conclusions from part (a) change now? What are the miss rates for the configurations above for the vector scaling operation? Which configuration has the best performance? Justify your answer. *[10 marks]*

3. (a) Discuss ways in which prefetching can improve performance and in which it can hurt performance. [*5 marks*]

(b) The compiler design team at ED Computers has asked your help to implement software prefetching in one particular piece of an application. The application core computes the final mark of all CS3 Computer Architecture (CAR) students at the University of Edinburgh. The database of students consists of a linked list of student objects. The object is shown below in Java:

```
class CARstudent {
  byte name[25];
  byte pracs;
  byte exam;
  byte finalmark;
  CARstudent next;
}
```

The main loop in the application is the following (assume that the initial value of *student* is loaded before the loop begins):

```
while (student) {
 student.finalmark = (byte) 0.25*student.pracs + 0.75*student.exam;
 student = student.next;
}
```

Each student object takes 32 bytes (25 for the name, 3 for the marks, and 4 for the pointer). Since the cache block size is also 32 bytes, only one student structure can be allocated per block. Considering only the loads and stores to the student object, what is the cache miss rate for the loop? [*5 marks*]

(c) Your job is now to eliminate the cache misses by inserting prefetches in the proper place and time. Show where in the loop code you would insert a prefetch operation. Assume that a cache miss, or a prefetch, takes 30 cycles to return the data. Also assume that each iteration of the loop takes approximately 20 cycles to execute, ignoring any cache misses. How many students would you need to prefetch in advance to eliminate all misses? [*10 marks*]

(d) Now consider the case where the student object is partitioned in two objects:

QUESTION CONTINUES ON NEXT PAGE

```
class Mark {
  byte pracs;
  byte exam;
  byte finalmark;
}

class CARstudent {
  byte name[24];
  Mark marks;
  CARstudent next;
}
```

Now each student object still takes 32 bytes (24 for the name and 8 for the two pointers). However, a mark object takes 3 bytes and cannot be allocated in the same cache block as the respective student object. The code now is changed to the following:

```
while (student) {
 student.marks.finalmark = (byte) 0.25*student.marks.pracs +
                           0.75*student.marks.exam;
 student = student.next;
}
```

Considering only the loads and stores to the student and mark objects, what is the cache miss rate for the loop now? Assuming that a prefetch takes 30 cycles to return the data and the loop executes in 20 cycles, show how you would insert prefetches in this case to eliminate all cache misses. [*5 marks*]