Chip Multiprocessors

Date: Wednesday 26th January 2011
Time: 14:00 - 16:00

Please answer any THREE Questions from the FOUR questions provided

Use a SEPARATE answerbook for each SECTION

For full marks your answers should be concise as well as accurate.
Marks will be awarded for reasoning and method as well as being correct

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
Section A

Note: When a question asks for instruction level code, a format similar to ARM assembler is expected. However, marks will not be lost if the format is incorrect as long as the meaning of the instruction is clear (from an accompanying explanation and/or comments).

1. Hardware Support for Synchronisation

   a) Using the instruction level code for a binary semaphore as an example, explain why it is necessary to provide support, in hardware, for synchronisation operations in a multi-core processor.

      (4 marks)

   b) Why is special hardware support not usually needed to support locking of resources shared between multiple processes which run on a uni-processor?

      (2 marks)

   c) Describe the ‘test and set’ instruction and explain why this provides a simple way of implementing synchronisation correctly.

      (4 marks)

   d) What are the major disadvantages of a ‘test and set’ instruction when considering implementation in a modern RISC processor.

      (2 marks)

   e) Explain how the ‘load linked/store conditional’ pair of instructions can be used to implement a binary semaphore and discuss how these instructions overcome the implementation problems of ‘test and set’. Your answer should include an indication of the code required at instruction level.

      (4 marks)

   f) The simplest form of barrier has a single variable initialised with a value N and has a ‘wait’ function which is called by a thread wanting to synchronise at the barrier. A thread calling ‘wait’ will decrement the variable and if the value is zero will immediately exit, otherwise it will wait at the barrier until the value is reduced to zero by other threads. Give instruction level code of the implementation of such a barrier using load linked and store conditional instructions and explain how it works.

      (4 marks)
2. **Cache Coherence**

a) Explain the cache coherence problem which exists in a shared memory multi-processor assuming that each core has its own local data cache.

   (4 marks)

b) Assuming that a cache line can be in one of three states M (modified), S (shared) or I (invalid) indicate, with the aid of diagrams, the valid states in which the same cache line can exist in two cores assuming a ‘snooping’ protocol.

   (4 marks)

c) Explain why these combinations of states are valid and any others are invalid stating any assumptions that you make.

   (6 marks)

d) Two cores are executing the simple program shown below which reads the variable x from memory, adds one to the value and writes it back to the same shared variable in memory. Assume that the execution is interleaved as shown with respect to any bus transactions which may occur. Time is flowing downwards. Assume a standard MSI protocol.

   Core 1       Core 2
   ldr r1,x     ldr r1,x
   add r1,r1,#1 add r1,r1,#1
   str r1,x     str r1,x

   Assuming that the value of x is not cached in either core at the start of the execution, list the cache states in each core, for the value of x after each instruction has completed execution and explain why these states occur.

   (4 marks)

e) If core 2’s cache entry for x were now to be flushed to main memory, what value would main memory contain? Is this correct?

   (2 marks)
Section B

3. **Transactional Memory**

   a) Explain briefly the role of Transactional Memory in a chip multiprocessor and outline what happens during a transaction.  
      (5 marks)

   b) Discuss the significance of the following issues and their implications for the implementation of Transactional Memory:

      i) eager validation versus lazy validation  
         (5 marks)

      ii) direct update versus deferred update  
         (5 marks)

   c) On a chip multiprocessor system with Transactional Memory, each thread contains the transaction below to swap integer elements of a shared array, a, where the elements to be swapped are indexed by variables i and j which are private to the threads.

      ```
      atomic {
        int temp = a[i] ;
        a[i] = a[j] ;
        a[j] = temp ;
      }
      ```

      Write equivalent code which uses fine-grained locking to achieve the same effect, as far as possible. Explain any difference in behaviour between the locking code and the code using a transaction.  
      (5 marks)
4. **Programming**

a) Java objects are defined to have locks, for use in synchronized statements and methods. Nevertheless there is an interface `Lock` in the package `java.util.concurrent.locks`. Using two distinct examples, explain why this interface (or a class implementing it) is needed. (6 Marks)

b) Parallel programs have been written using OpenMP, MPI, and CUDA. Using simple examples where appropriate, compare and contrast these different notations from the point of view of the abstract model they provide to the programmer. (9 marks)

c) Why is the functional programming paradigm attractive as a way of programming chip multiprocessors? Why is it not the (whole) solution? (5 marks)

END OF EXAMINATION