COMPUTER SCIENCE TRIPOS Part IB

Wednesday 8 June 2011 1.30 to 4.30

COMPUTER SCIENCE Paper 5

*Answer **five** questions.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags

SPECIAL REQUIREMENTS
Approved calculator permitted

1 Computer Design

Consider the following UltraRISC processor which has just one instruction (so there is no opcode), with one operand (an address).

```
module UltraRISC();
   logic [7:0] mem[31:0]; // memory
   logic [7:0] pc;  // program counter
   logic [7:0] ir;  // instruction register
   logic [7:0] acc; // accumulator
   logic       step;
   logic       clk;
   logic [7:0] next_pc, next_ir;
   logic [8:0] next_acc;
   logic       borrow;
   parameter Lt=16, Lx=17, Ly=18, Lz=19, Lstop=31;
   initial begin
      clk <= 1;  pc <= 0;  step <= 0;  acc <= 0;
      // Code               Data
      mem[ 0] <= Lt;        mem[Lt] <= 13; // holds T
      mem[ 1] <= Lt;        mem[Lx] <= 13; // holds X
      mem[ 2] <= Lt;        mem[Ly] <= 7;  // holds Y
      mem[ 3] <= Lx;        mem[Lz] <= 3;  // holds Z
      mem[ 4] <= Lx;        mem[Lstop] <= 0;
      mem[ 5] <= Ly;
      mem[ 6] <= Lt;
      mem[ 7] <= Lt;
      mem[ 8] <= Lx;
      mem[ 9] <= Lt;
      mem[10] <= Lt;
      mem[11] <= Lt;
      mem[12] <= Lz;
      mem[13] <= Lx;
      mem[14] <= Lstop;
      mem[15] <= 0;
   end // initial begin
   always #5 clk <= !clk;
```

[**continued. . .**

```
   always_comb
      if(step==0) begin
         next_ir = mem[pc];
         next_acc = acc;
         next_pc = pc+1;
      end else begin
         next_ir = ir;
         next_acc = mem[ir]-acc;
         borrow = next_acc[8];
         next_pc = pc+borrow;
      end

   always_ff @(posedge clk) begin
      step <= !step;
      ir <= next_ir;
      pc <= next_pc;
      acc <= next_acc;
      if(step) mem[ir] <= next_acc;
      if(ir==Lstop) begin
         $display("result = %d, finished",acc);
         $finish;
      end
   end
endmodule
```

(*a*) What is the CPI (cycles per instruction) for this processor? [3 marks]

(*b*) What function does the one instruction perform? [5 marks]

(*c*) What result is produced when the program held in `mem` is executed? Explain your answer. [10 marks]

(*d*) How does the code density compare with the MIPS32 ISA? [2 marks]

2 Computer Design

(*a*) For a processor pipeline, what is the difference between control and data dependencies? [4 marks]

(*b*) For a superscalar processor, under what conditions can instructions from the same thread be executed in parallel? [4 marks]

(*c*) What does it mean to have coherent shared memory and in this context what do the terms *write propagation* and *write serialisation* mean? [8 marks]

(*d*) If a simultaneous multithreading processor core supports two hardware threads (or *Hyperthreads*), under what conditions might performance be decreased by scheduling two jobs in parallel rather than running them sequentially? [4 marks]

3 Computer Design

(*a*) Why do modern processors need caches and what statistical properties do they exploit to improve processor performance? [5 marks]

(*b*) How are cache tags used in direct-mapped and fully-associative caches? [5 marks]

(*c*) A translation look-aside buffer (TLB) is a form of cache, but what does it store and how are TLB misses handled? [5 marks]

(*d*) A processor architect proposes the use of a large direct-mapped TLB rather than the traditional small fully-associative TLB. What might the performance implications be? [5 marks]

4 Computer Networking

The popular press suggest that the Internet is a great success.

Based on the range of topics covered in the Computer Networking course, critique the technological success and failure of the Internet. Assertions alone will not constitute an answer to the question: please supply evidence by examples.

[20 marks]

5 Computer Networking

Consider two physically-separated entities **A** and **B**. **B** has been supplied messages that will be sent to **A** following these conventions:

- **A** gets a request from the layer above to retrieve the next data ($\mathcal{D}$) message from **B**.
- **A** must send a request ($\mathcal{R}$) message to **B** on the **A**-to-**B** channel.
- Upon receipt of an $\mathcal{R}$, **B** will send $\mathcal{D}$ back to **A** on the **B**-to-**A** channel.
- **A** should deliver exactly one copy of each $\mathcal{D}$ message to the layer above.
- $\mathcal{R}$ messages may be lost (but will not be corrupted) in the **A**-to-**B** channel.
- $\mathcal{D}$ messages are always delivered correctly (no loss or corruption).
- The delay along each channel is unknown and variable.

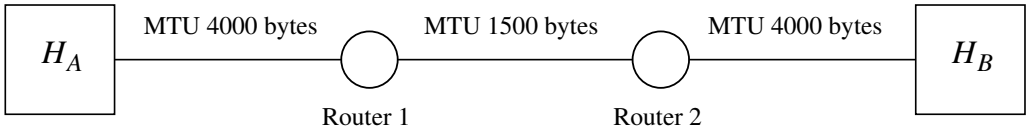
Give the FSM describing a protocol employed by **A** and **B**.

This FSM must compensate for the loss-prone channel between **A** and **B**, as well as implementing message passing to the layer above at entity **A**. Your FSM must not use more mechanisms than is necessary.

[20 marks]

6 Computer Networking

(*a*) The diagram below shows a TCP connection between Hosts H_A and H_B passing through networks with different values of Maximum Transmission Unit (MTU) shown. Version 4 of the Internet Protocol (IPv4) is in use.



H_A chooses a TCP segment size of 3000 bytes of data (TCP and IP headers are each 20 bytes in size).

(*i*) Describe the way in which fragmentation takes place as H_A sends data to H_B. Include the arithmetic used to calculate fragment sizes. Explain the saving that may be made by H_A choosing an optimal TCP segment size when sending 60KBytes of data. [8 marks]

(*ii*) Briefly explain how the situation described in part (*i*) would be handled if Internet Protocol version 6 (IPv6) were used. [2 marks]

(*b*) The formulae below are used in TCP implementations to compute a value for the retransmission time-out ($\mathcal{R}$). R is an estimate of the round-trip time, M is the most recently measured round-trip measurement, $\alpha = 0.875$ and $h = 0.25$.

$$\begin{aligned} D &\leftarrow D + h(|M - R| - D) \\ R &\leftarrow \alpha R + (a - \alpha)M \\ \mathcal{R} &= R + 4D \end{aligned}$$

(*i*) How is M measured? [2 marks]

(*ii*) Explain the principles behind the design of these formulae and the values h, α and D. [8 marks]

7 Concurrent and Distributed Systems

Database concurrency control enables multiple transactions to proceed simultaneously against a shared database.

(*a*) Give an example each of *conflicting* and *non-conflicting* operations on objects in a database. [2 marks]

(*b*) Explain how timestamp ordering (TSO) works, and how it responds to conflicts between transactions. [4 marks]

(*c*) Explain cascading aborts, and use an example to show how they can occur with TSO. [4 marks]

(*d*) Why is TSO not subject to deadlock? [2 marks]

(*e*) Explain why TSO distributes well, compared with two-phase locking. [4 marks]

(*f*) Construct a workload that performs badly under TSO, and explain why it does so. What could you do to improve its performance? [4 marks]

8 Concurrent and Distributed Systems

(*a*) Consider a queue data structure, with the following interface:

```
interface Queue {
  void push(Object val);
  Object pop();
};
```

(*i*) Write pseudocode for a semaphore-based implementation of `Queue`. Your implementation should allow concurrent push and pop when it is safe to do so, but not when it is unsafe. The queued data should be stored in an array of fixed length n. [6 marks]

(*ii*) Explain the specific circumstances under which concurrent push and pop is unsafe. Explain how your solution in part (*i*) addresses these. [2 + 2 marks]

(*b*) You decide to use this data structure to manage a service, where people around the world push jobs of work to be done, and others pop jobs and do them. (This is a description of Amazon's Mechanical Turk.)

This could be implemented as a centralised service, offered through Object-Oriented Middleware.

(*i*) How can the Object-Oriented Middleware maximise the utilisation of this service? [2 marks]

(*ii*) Object-Oriented Middleware makes the calls to push and pop look local. Name **two** things that this hides from the programmer using the service and say why each is a problem. [4 marks]

(*iii*) What aspects, that are not part of the middleware, might hinder scalability in terms of the number of potential pushers and poppers? [4 marks]

9 Concurrent and Distributed Systems

(*a*) We have considered four types of middleware: remote procedure call, object-oriented middleware, message-oriented middleware, and event-based middleware.

(*i*) Each middleware has a core action, such as a remote procedure call or a remote method invocation. This entails data transfer that is either unidirectional (out of or into the calling context) or bidirectional (in and out). State which is used by each of the **four** types of middleware. [4 marks]

(*ii*) Does each of these—uni- and bidirectional data transfer—have sufficient expressive power for programming? Explain your answer. [2 marks]

(*iii*) One of the characteristics of distributed systems is that they lack global time. Given your answers above, what effect might this have on middleware use? [4 marks]

(*b*) (*i*) What are *causal* and *totally ordered* message delivery? [2 marks]

(*ii*) Which does vector clocks provide? [1 mark]

(*iii*) The vector clock algorithm is a way of sharing state, ensuring that every process knows what it needs to about how far the others have progressed. Why is it critical that messages having vector timestamps are never lost? [2 marks]

(*c*) (*i*) Storage services can be *stateful* or *stateless*. Give **one** advantage and **one** disadvantage of each. [2 marks]

(*ii*) If you were designing a service to support film production, which would you use and why? [3 marks]

END OF PAPER