ENGINEERING TRIPOS PART IIA

ELECTRICAL AND INFORMATION SCIENCES TRIPOS PART I

Friday 7 May 1999 9 to 12

Paper E6

COMPUTING SYSTEMS

Answer not more than **five** *questions.*

All questions carry the same number of marks.

The **approximate** *number of marks allocated to each part of a question is indicated in the right margin.*

(TURN OVER

1 (a) Describe briefly the design of a 4-bit *carry-lookahead adder*. Explain why larger adders typically employ several levels of carry-lookahead, and discuss the factors affecting the choice of the optimal number of levels. What are the asymptotic time and space requirements of a carry-lookahead adder (i.e. how do the time and space requirements grow with n, for an n-bit adder where n is large)? [8]

(b) Figure 1 shows an alternative design known as a *carry-select adder*. The principle of operation is that the hardware for each block is duplicated and two additions are performed in parallel, one assuming the carry-in to the block is 0 and the other assuming the carry-in is 1. When the result of the previous block is known, the correct sum is simply selected using the multiplexors (denoted by MX in Fig. 1). Note that most of the individual bits of the two numbers to be added are not shown in Fig. 1 for clarity.

(i) If each block uses ripple-carry internally, so that the time taken for a k-bit block to add is k time units, and if it takes one time unit to compute the multiplexor inputs (c8 and c13) from the AND/OR gate inputs, explain why, for optimal performance, most of the blocks are one bit longer than their immediate predecessors. Sketch an optimal design for a 16-bit carry-select adder. [2]

(ii) Hence deduce the asymptotic time and space requirements of a carry-select adder. [6]

(c) Comment on the practical significance of asymptotic behaviour when designing real adders. [4]

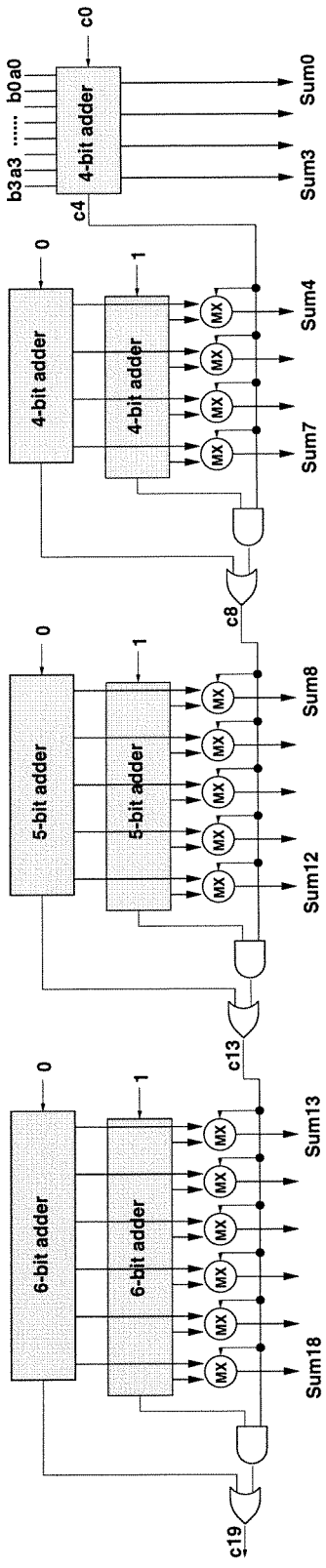

Fig. 1

(TURN OVER

2 (a) What are the properties of typical computer programs that motivate the inclusion of a cache between the CPU and main memory? [2]

(b) A particular CPU can address 2^{32} bytes of physical memory through a 128 KByte cache. The cache uses 4-word blocks and is 8-way set-associative. Show how the physical address can be broken up into fields and explain the role played by each field in a typical cache access. Be sure to state the length (in bits) of each field. [4]

(c) What are the two principal ways in which a cache can be redesigned to reduce the miss rate? Explain why neither of these refinements will necessarily make the computer go faster. [4]

(d) Another way to reduce the cache miss rate is to optimise the software running on the computer. Figure 3 contains the obvious Pascal code to multiply two 1000×1000 matrices together. The matrix `c` is multiplied by the matrix `b`, and the result stored in the matrix `a`. Figure 4 contains an optimised version of the code, which uses a technique called *blocking* to reduce the cache miss rate: in this example the *blocking factor* is 10. The relevant data definitions and declarations for both versions of the code can be found in Fig. 2.

Consider running the two code segments on a machine with a 1 KByte fully-associative cache, 1-word blocks and least recently used (LRU) block replacement.

(i) If a `Real` number requires 4 bytes of storage, how many matrix elements can fit in the cache? [1]

(ii) Estimate the total number of cache misses for the version of the code in Fig. 3. [4]

(iii) Estimate the total number of cache misses for the version of the code in Fig. 4. [5]

```
CONST
   matrixSize   = 1000;
   blockingFactor = 10;
TYPE
   Matrix = ARRAY[1..matrixSize] OF ARRAY[1..matrixSize] OF Real;
VAR
    a, b, c          : Matrix;
    i, j, k, jj, kk : Integer;
    r                : Real;
```

Fig. 2

```
FOR i := 1 TO matrixSize DO
   FOR j := 1 TO matrixSize DO BEGIN
      r := 0.0;
      FOR k := 1 TO matrixSize DO
         r := r + b[i][k] * c[k][j];
      a[i][j] := r;
   END;
```

Fig. 3

```
jj := 1; kk := 1;
WHILE (jj <= matrixSize) DO BEGIN
   WHILE (kk <= matrixSize) DO BEGIN
      FOR i := 1 TO matrixSize DO
         FOR j := jj TO (jj + blockingFactor - 1) DO BEGIN
            r := 0.0;
            FOR k := kk TO (kk + blockingFactor - 1) DO
               r := r + b[i][k] * c[k][j];
            a[i][j] := a[i][j] + r;
         END;
      kk := kk + blockingFactor;
   END;
   kk := 1;
   jj := jj + blockingFactor;
END;
```

Fig. 4

(TURN OVER

3 In your role as an international software engineering troubleshooter, you have been called in to investigate reported problems in a bank's *debit card server* software, originally developed in the late 1980s.

Figure 5 shows extracts from the requirements definition. Figure 6 shows an extract from the source code. The source code is written in a multi-threaded (concurrent) version of Pascal.

(a) With reference to the *concurrency* and *lifetime* requirements, describe two *features* of the implementation that could cause erroneous program behaviour. What are the consequences for the bank in both cases? [8]

(b) How would you enhance the implementation to solve both of these problems? [8]

(c) Describe how testing should have detected the lifetime problem. [4]

```
2.4 Debit Operation
2.4.1 Any account holder can use their card to purchase items in shops.
This constitutes a debit operation.
The point-of-sale terminal transmits the relevant information
(such as card number and debit amount) to the debit card server.

2.4.2 The expiry date of the card must be checked to ensure the card
is still valid

2.4.3 The amount of money in a user's account is queried from the underlying
database system. The debit operation can only proceed if the current balance
is more than or equal to the amount to be debited.
...

3.4 Lifetime Requirements
3.4.1 The software system must operate correctly at least until the year 2010.
...

3.6 Concurrency Requirements:
3.6.1 The server software system must be capable of handling operations
concurrently, and able to support up to 500 concurrent client requests.

3.6.2 Joint account holders and business account holders have 2 or more
cards, each of which can be used concurrently to debit the same account.
```

Fig. 5

(cont.

```
TYPE return_code = (Success, Failure);
TYPE card_type = ARRAY[0..12] OF INTEGER
TYPE card_account = RECORD
      card_number: card_type;
      expiry_month: 1..12;
      expiry_year: 0..99;
      current_balance: INTEGER;
END;

PROCEDURE debit(card: card_type, amount_to_debit: INTEGER,
                  VAR return: return_code)
VAR
     acc: card_account;
     this_month, this_year : INTEGER;
     new_balance : INTEGER;

BEGIN

     (* Read card details from DB and store in acc *)
     retrieve_card_details(card, acc);

     (* Check the card has not expired *)
     get_current_date(this_month, this_year);
     IF ((current_year < acc.expiry_year) OR
         ((current_year = acc.expiry_year) and
          (current_month <= acc.expiry_month)))THEN
     BEGIN
          (* Check there is enough money in the account *)

          new_balance := (acc.current_balance  amount_to_debit);
          IF (new_balance > 0) THEN
          BEGIN
              (* Store the new balance in the database *)
              write_new_balance(card, new_balance);
              return := Success;
          END
          ELSE
               return : = Failure;
          END
          ELSE
               return := Failure;

     END (* debit *);
```

Fig. 6

(TURN OVER

4 (a) Define what is meant by a *hard real-time system.* Which of the following software systems would you consider as hard real-time:

(i) a fly-by-wire control system for aircraft;

(ii) the London Ambulance Service emergency incident system;

(iii) a nuclear reactor monitoring system, and

(iv) an office information system? [5]

(b) Is it likely that a *prototyping* approach to requirements capture would be used in the development of a hard real-time system? Explain your answer. [5]

(c) What is the purpose of a software *requirements specification*? Why do many procurers of hard real-time systems insist on a formal specification? [5]

(d) Why is a multi-purpose operating system like Windows-NT **not** appropriate to support hard real-time systems? [5]

5 (a) Consider a pattern classification task involving C classes, $\omega_1 \ldots \omega_C$, from a feature vector, $\mathbf{x}$.

(i) Write down Bayes' formula for computing the *posterior* probabilities for each class and explain briefly the significance of each term and how they might be estimated in practice.

(ii) Write down Bayes' decision rule for minimising the average probability of classification error. [8]

(b) In a two-class pattern classification problem, using two-dimensional features, the individual classes, denoted by ω_1 and ω_2, have features with class conditional probability densities which have Gaussian distributions given by:

$$p(\mathbf{x}|\omega_i) = \frac{1}{2\pi|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T \Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)\right\}.$$

where superscripts T and -1 respresent the transpose and inverse of a vector and matrix respectively. The means of the two classes are

$$\mathbf{m}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{m}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

and the corresponding covariance matrices are given by

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

(i) Taking the *prior* probabilities to be $P(\omega_1) = 0.6$ and $P(\omega_2) = 0.4$, derive the optimal decision rule for this problem. State any assumptions that you have made.

(ii) Compute and sketch the decision boundary. What was the effect of the prior probabilities? [8]

(c) Describe the use of Receiver Operating Characteristic (ROC) curves in pattern classification problems. Suggest a way of deriving an ROC curve for the problem described above. [4]

(TURN OVER

6 A two-class pattern classification problem is defined by a set of N labelled training data,

$$\{\mathbf{x}_i,\ f_i\}, \quad i = 1, ..., N.$$

where $\mathbf{x}$ is a d-dimensional feature vector and f indicates the class membership.

(a) Explain how the parameters of a *linear discriminant function*, $g(\mathbf{x})$, can be estimated from the training data by treating the classification problem as a multi-dimensional interpolation problem. [4]

(b) Derive, from first principles, the solution for the parameters of the classifier as the pseudo-inverse of a matrix containing the feature vectors. [6]

(c) Describe an iterative method to find this solution. What factors affect convergence of the solution? [5]

(d) Explain clearly how the *perceptron algorithm* differs from the linear discriminant derived above. Use sketches of data distributions in one (or two) dimensions to illustrate your answer. [5]

7 (a) What is meant by a *well-defined problem* in Artificial Intelligence? Describe how a solution can be found by state-space search. [5]

(b) Describe the A* search algorithm and compare its computational complexity and performance with *depth-first* and *breadth-first* search techniques. [7]

(c) Figure 7 shows a triangular object at a start position S on a plane which has rectangular obstacles (shown shaded). Consider the problem of moving this object from S to the goal position G. The *configuration space* representation is also shown (with labelled vertices) for the case in which the triangular object is allowed to translate.

(i) Explain briefly the advantages of this representation.

(ii) If the *successor* function explores the vertices alphabetically, describe the solution that would be found by depth-first, breadth-first and A* search. Give details of any *evaluation* functions used. You should **not** generate a search tree.

(iii) How would the solution found by the A* search algorithm change if rotation of the triangular object is allowed? [8]

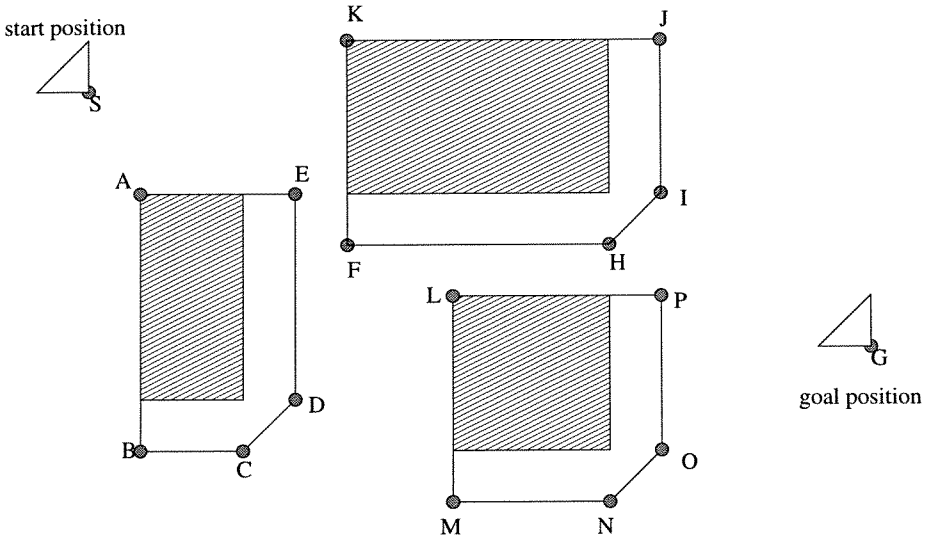

Fig. 7

(TURN OVER

8 (a) Explain briefly how *propositional* logic can be used to represent and reason about knowledge in a computer tractable form. How is *refutation* or *proof by contradiction* used in making inferences. [6]

(b) List the truth table for the logical connective $P \rightarrow Q$. Hence verify the *soundness* of the following inference rule:

$$((A \rightarrow B) \wedge (A \vee C)) \rightarrow (B \vee C).$$

[6]

(c) A database of family relationships includes the following facts in *clause* form:

```
son(Edward, Henry)
daughter(Elizabeth, Henry)
daughter(Mary, Henry)
daughter (Elizabeth, Anne)
son(Charles, Elizabeth)
```

where the predicate `son(X,Y)` represents the fact that X is the son of Y.

(i) Add a general rule to the database that defines a sibling (brother or sister).

(ii) Convert the clauses and rules to *conjuctive normal form.*

(iii) Use *resolution theorem proving* to find the siblings of Elizabeth. [8]

END OF PAPER