UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2003

BEng Honours Degree in Computing Part III
MEng Honours Degree in Electrical Engineering Part IV
BEng Honours Degree in Information Systems Engineering Part III
MEng Honours Degree in Information Systems Engineering Part III
BSc Honours Degree in Mathematics and Computer Science Part III
MSci Honours Degree in Mathematics and Computer Science Part III
MSci Honours Degree in Mathematics and Computer Science Part IV
MSc in Advanced Computing
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute
This paper is also taken for the relevant examinations for the Associateship of the Royal College of Science

PAPER C332=I3.26=E4.31

ADVANCED COMPUTER ARCHITECTURE

Tuesday 29 April 2003, 14:30
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1 This question concerns cache consistency in the IBM Power4 processor, as described in the paper "POWER4 System Architecture" (Tendler et al, IBM Journal of Research and Development, V46 No.1, Jan 2002), which you should have available to you in the examination. **See, in particular, pages 15-19.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a What is stored in the L2 cache directory?

b What causes invalidation of data in the L1 cache (that is a "back invalidate")? How is inclusivity information used?

c The 32-MByte L3 cache is 8-way set-associative with 512-byte cache lines managed as four 128-byte sectors. Physical addresses are 42 bits. How big is the L3 cache directory?

d How big would the L3 cache directory be if the L3 cache line size were 128 bytes (that is, if it were not sectored)?

e Would a non-sectored L3 cache design lead to a lower average memory access time? Explain your reasoning.

(The five parts carry, respectively, 25%, 20%, 25%, 15% and 15% of the marks).

2 This question concerns branch prediction in the IBM Power4 processor, as described in the paper "POWER4 System Architecture" (Tendler et al, IBM Journal of Research and Development, V46 No.1, Jan 2002), which you should have available to you in the examination. **See, in particular, pages 8 and 9.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a Give an example (in a suitable high-level language) in which the POWER4's global history table would mispredict frequently, but the local predictor would achieve better branch prediction. Briefly, explain why.

b How many bits of the instruction address would influence the prediction outcome in a 16K-entry *gselect(11,1)* predictor?

c This question concerns the POWER4's local predictor. Many benchmark results (eg those presented in the textbook) show that a Branch History Table with just one bit per entry performs poorly, but using two bits (or perhaps more) per entry dramatically improves performance.

(a) Why might this be so?

(b) Why did the POWER4 designers choose just one bit per entry?

d The POWER4's 32-entry Branch Target Buffer is called the "count cache". It is used for indirect branch and procedure call instructions (where the branch destination is given in a register).

The C/C++/Java "switch" statement allows control flow to be selected from a number of alternatives. A common use might be in a bytecode interpreter:

```
switch (opcode) {
case 1: /* code for when opcode==1 */
case 2: /* code for when opcode==2 */
...
case 64: /* code for when opcode==64 */
}
```

This can be implemented either using a sequence of conditional branches, or using an indirect branch. What determines whether it is better to use an indirect branch?

(The four parts carry, respectively, 10%, 15%, 25%%, and 50% of the marks).

3 This question concerns dynamic instruction scheduling and speculative execution in the IBM Power4 processor, as described in the paper "POWER4 System Architecture" (Tendler et al, IBM Journal of Research and Development, V46 No.1, Jan 2002), which you should have available to you in the examination. **See, in particular, pages 11-14.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a Consider the following events which may occur during processing of an instruction (or IOP):

(i) Fetched
(ii) Dispatched
(iii) Issued
(iv) Committed
(v) Rejected
(vi) Flushed

For each event, write down the number of instructions involved. If necessary, explain your answer very briefly.

b The following loop is shown in MIPS assembly code for your convenience. It computes the sum of a large vector of double-precision (8-byte) floating point numbers:

```
$L5:
        l.d     $f4,0($4)     # vector base address in reg $4
        addu    $5,$5,-1      # no of elements in reg $5
        add.d   $f2,$f2,$f4   # sum into $f2
        addu    $4,$4,8
        bne     $5,$0,$L5     # (POWER4 branches are not delayed)
                              # on exit result is in $f2
```

Suppose this loop were executed on the POWER4, and that each MIPS instruction corresponds to one POWER4 IOP.

(i) Estimate the number of cycles per 1000 iterations for this loop (note that the floating point units are fully-pipelined but take 6 cycles to complete).
(ii) Estimate the number of instructions per clock cycle (IPC).
(iii) Briefly explain how the loop could be modified to improve the performance (you do not need to show detailed code - just explain the principle).
(iv) Estimate the number of cycles per 1000 iterations that could be achieved using your restructured loop.
(v) Suppose the vector is too large to fit in L1 or L2 cache, but is available in L3 cache. Is the bottleneck for this loop in the central processor, or in the memory subsystem? Discuss what architectural features determine your answer.

(The two parts carry, respectively, 50% and 50% of the marks).

4 In this question, consider the following loop:

```
declare float U[0:M, 0:N]

for t = 1 to M do
  for i = 1 to N-1 do
S:  U[t,i] = (U[t-1,i-1] + U[t-1,i+1]) * 0.5
```

a Suppose `M=4` and `N=6`. Draw the iteration space graph for the loop. Mark on the graph all the dependences present.

b Write down all the dependences present in the loop. For each dependence, indicate whether it is a data-dependence or an anti-dependence, whether it is loop carried, and write down its dependence distance vector.

c Write down a unimodular transformation matrix which represents a valid interchange of the two loops.

d Draw the iteration space for the loop above after this transformation has been applied. Show the dependences.

e Can the transformed loop be tiled? Justify your answer with reference to your iteration space graph.

f Suppose now that `M=4` and `N` is large. Given a fully-associative data cache with enough space for 128 `doubles`, what is the largest tile size which still makes efficient use of the cache?

g Explain, using a diagram, what could go wrong if the cache is direct-mapped instead of fully-associative.

(The seven parts carry, respectively, 15%, 20%, 10%, 15%, 10%, 20% and 10% of the marks).

End of Paper