UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1997

BEng Honours Degree in Computing Part III
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
MEng Honours Degree in Electrical and Electronic Engineering Part IV
MSc Degree 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
Diploma of Membership of Imperial College
Associateship of the City and Guilds of London Institute
Associateship of the Royal College of Science

PAPER 3.32 / I3.20 / E4.31

PARALLEL ARCHITECTURES

Tuesday, April 22nd 1997, 10.00 - 12.00

Answer THREE questions

For admin. only: paper contains 4 questions

1 This question concerns the implementation of the following loop on a shared-memory multiprocessor:

```
for i1 = 1 to N do
 for j1 = 1 to N do
  for i2 = 1 to N do
   for j2 = 1 to N do
    if (A[i1,j1] > 0) and (A[i2,j2] > 0)
    then
     m = classify(100*((j2-j1)/(i2-i1)); /* (nearest int within bounds of 'line') */
     c = classify(100*(i1 - m*j1));      /* (nearest int within bounds of 'line') */
     line[m,c] = line[m,c] + 1;
    endif
   enddo
  enddo
 enddo
enddo
```

Assume that the number of processing elements is small compared to N, and that an invalidation-based cache coherence protocol is used.

a Is loop interchange valid here? Explain.

b What synchronisation issues arise in implementing this loop on a shared-memory multiprocessor?

c How can load imbalance be avoided without excessive management overheads?

d What aspects of the memory/cache architecture might influence the performance of a parallel implementation of this program on a coherent-cache shared-memory multiprocessor? Consider each of the memory references in the program in turn.

e *Briefly sketch* how this loop might be implemented efficiently on a coherent-cache shared-memory multiprocessor.

(The five parts carry, respectively, 10%, 20%, 20%, 20% and 30% of the marks).

2 Consider the following loop:

```
for i = 1 to N do
  for j = 1 to N do
    A[i+1,j+1] := A[i,j] + A[i+1,j] + A[i,j+1]
```

a Estimate the number of cache misses per iteration when executing the original version of the loop.

b Suppose the CPU has a small, write-back data cache with block size of two words. Estimate the number of cache misses per iteration when executing the original version of the loop. Take care to state any assumptions you need to make.

c Draw a diagram showing the iteration space for this loop, and the dependences between loop iterations.

d Demonstrate by drawing an appropriate diagram that the following blocked implementation of the loop is valid:

```
for ii = 1 to N step B do
  for jj = 1 to N step B do
    for i = ii to ii+B-1 do
      for j = jj to jj+B-1  do
        A[i+1,j+1] := A[i,j] + A[i+1,j] + A[i,j+1]
```

e Suppose the CPU has a 1K-word (8KByte) direct-mapped data cache with block size of *one* word. Estimate the number of cache misses per iteration when executing the blocked implementation of the loop using a suitable value for B. Take care to state any assumptions you need to make.

(The five parts carry, respectively, 10%, 20%, 30%, 20%, and 20% of the marks).

Turn over ...

3 Consider the two following processes, running in parallel on a shared-memory multiprocessor:

```
/* processor 1 */
A := 0;
:
A := 1;
if (B == 0) {
  P();
}
```

```
/* processor 2 */
B := 0;
:
B := 1;
if (A == 0) {
  P();
}
```

Assume that procedure P is not called from elsewhere.

a Assume that the processes are running on an idealised shared-memory machine.
Can both processors execute P simultaneously? Explain your answer carefully.

b Assume that the two fragments are running on different CPUs in a coherent-cache shared-memory multiprocessor, using an invalidation protocol.
Can both processors execute P simultaneously? Explain the potential problem carefully.
What precautions are needed to preserve the expected behaviour?

c Assume that the two fragments are running on different CPUs in a coherent-cache shared-memory multiprocessor, using an *update* protocol.
Can both processors execute P simultaneously? Explain your answer carefully.
What precautions are needed to preserve the expected behaviour?

d Some multiprocessors do not guarantee the expected behaviour for this example, but instead guarantee consistency only when CPUs synchronise (e.g. after passing through a LOCK or BARRIER). Explain why this might offer a performance advantage.

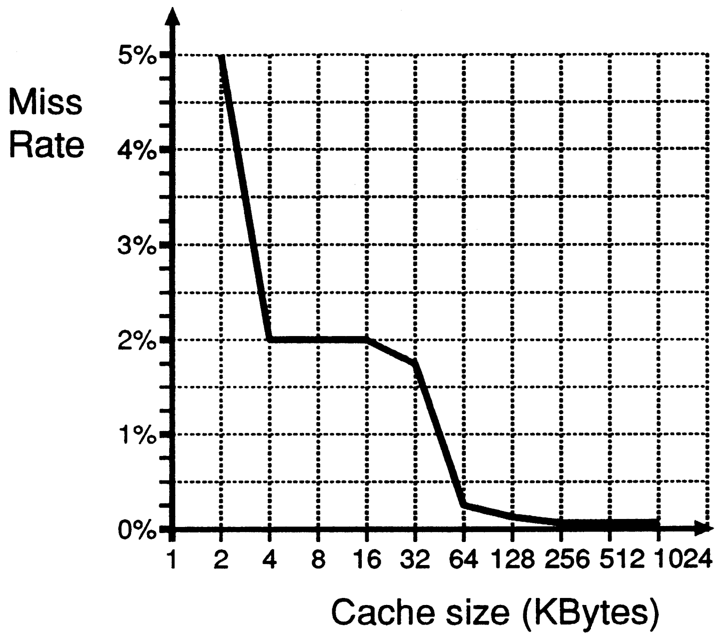
e Consider an 8-CPU coherent-cache shared-memory multiprocessor, using an invalidation protocol, in which directories (which indicate which caches hold a copy of a particular value) are represented using a singly-linked sharing list. Suppose the CPUs are interconnected using a full crossbar network.

i What is the minimum number of messages needed to perform an invalidation? Explain your answer carefully.

ii What is the maximum (i.e. worst-case) number of messages needed to perform an invalidation? Explain your answer carefully, using a diagram if necessary.

(The six parts carry, respectively, 15%, 20%, 10%, 10%, 15% and 30% of the marks).

4 The graph below shows the approximate cache miss rate for a certain application, for increasing cache size:



Suppose the job is running on *one* processor of a two-processor shared-memory machine which uses a straightforward bus-based invalidation cache coherency protocol, with a cache for each processor, and a cache line size of 16 bytes. Assume the following:

1 25% of instructions are loads or stores.
2 The CPI assuming no cache misses is 1.5
3 The clock rate is 300MHz (clock period 3.33ns)
4 A cache miss takes 40 cycles (133ns)

a Calculate the MIPS rate of this application assuming no cache misses.

b Explain *two different* ways by which it might be possible to improve performance while reducing the MIPS rate.

c Estimate the CPI the system will achieve on this application for cache sizes of 4KB, 32KB and 64KB.

d Suppose that the operating system's scheduler decides to stop the job, and restart it on the other CPU. To do this involves a management overhead of 40μs (to save the registers of the process, and to restart the job on the new CPU by setting up the registers).
Suppose that the new CPU's cache contains no useful data, so the restarted job will suffer cache misses which would not have occured if the job had not been migrated. *Estimate* the total delay attributable to additional cache misses each time the job is migrated, for the three cases when cache size is 4KB, 16KB and 64KB.

(The four parts carry, respectively, 10%, 20%, 30% and 40% of the marks).

End of Paper