1 This question concerns out-of-order execution in the IBM Power5 processor, as described in the paper "IBM Power5 Chip: A Dual-core Multithreaded Processor" (IEEE Micro March-April 2004), which you should have available to you in the examination. **See, in particular, pages 43–44.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a The PowerPC instruction set has 32 integer ("general-purpose") registers and 32 floating-point registers. The Power5 has 120 physical general-purpose registers and 120 physical floating point registers. Suppose the instruction set were extended to support 120 logical registers. Give *two* reasons why register-renaming ("mapping") might still be useful.

b What data structure is used at instruction dispatch to determine the physical source registers for an instruction?

c How is the mapping from a thread's logical to physical registers modified when a branch misprediction is discovered?

d How is the GCT (global completion table) affected when a thread suffers a branch misprediction?

e Physical registers are allocated dynamically. When are they allocated? What conditions determine when a physical register can be freed for re-allocation?

f The Power5 maintains a load reorder queue and a store re-order queue, to detect out-of-order execution hazards. When do you think this mechanism is used? Under what conditions would this mechanism delay execution of a load?

The six parts carry, respectively, 20%, 10%, 10%, 10%, 35%, and 15% of the marks.

2 This question requires access to the paper “IBM Power5 Chip: A Dual-core Multithreaded Processor” (IEEE Micro March-April 2004), which you should have available to you in the examination. Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

This question concerns an alternative to the Power5 design philosophy (based loosely on the CELL POWERPC Processing Element). Assume, for the purposes of this question, the following architectural features:

- In-order, dual-issue pipeline
- Integer pipeline with 11 stages
- One fully-pipelined floating-point add/subtract unit
- One fully-pipelined floating-point multiply unit
- Two-way simultaneous multithreading (SMT)
- L1, L2, L3 caches and memory system similar to Power5
- A tournament branch predictor like the Power5’s
- Clock rate twice as fast as the Power5

a Suggest a sensible structure for the integer pipeline. Explain briefly the function of each of the 11 stages. Take care to include how the branch predictor is accessed.

b Identify *two* forwarding paths required to avoid unnecessary stalls in your pipeline.

c Give three circumstances in which instruction issue might stall in this design.

d This design supports two-way simultaneous multithreading. At which pipeline stage does this processor select between the two threads? What information is required to make this decision?

e Outline briefly the characteristics of an application program that would run faster on the Power5 than on this design. Justify your answer carefully.

The five parts carry, respectively, 45%, 15%, 20%, 10%, and 10% of the marks.

3 This question partly concerns multithreading in the IBM Power5 processor, as described in the paper "IBM Power5 Chip: A Dual-core Multithreaded Processor" (IEEE Micro March-April 2004), which you should have available to you in the examination. **See, in particular, page 45.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

Consider the following fragment of application code:

```
double A[N][N], B[2*M][2*M], C[N][N];

for (int i=M; i<=N-M; i++)
  for (int j=M; j<=N-M; j++)
    for (int m=-M; m<M; m++)
      for (int n=-M; n<M; n++)
        C[i][j] += A[i+m][j+n]*B[M+m][M+n];
```

(Assume that $N \geq 2M$).

a Can the outermost loop be executed in parallel? Justify your answer carefully.

b Can the innermost loop be directly vectorised for execution on a machine with vector instructions, such as the Cray-1 or DLXV?

c Consider the following variant of the code above:

```
for (int i=M; i<N-M; i++)
  for (int m=-M; m<M; m++)
    for (int n=-M; n<M; n++)
      for (int j=M; j<N-M; j++)
        C[i][j] += A[i+m][j+n]*B[M+m][M+n];
```

This version was derived from the original loop nest shown at the start of the question by interchanging loops. Write down a unimodular transformation matrix that represents this transformation.

d On many machines this version is considerably faster (50%–500%) for most values of `N` and `M`. Can you explain why? Try to offer at least two reasons.

e The outermost loop of the improved version could be split into two, and executed by two concurrent threads sharing the same Power5 processor core. Comment on whether this is likely to result in a speedup.

The five parts carry, respectively, 15%, 15%, 15%, 30%, and 25% of the marks.

4 This question concerns multithreading in the IBM Power5 processor, as described in the paper "IBM Power5 Chip: A Dual-core Multithreaded Processor" (IEEE Micro March-April 2004), which you should have available to you in the examination. **See, in particular, pages 44–46.** Where the paper is incomplete, you are invited to speculate using your understanding of the underlying architectural principles.

a In single-threaded mode, under what circumstances might instruction dispatch be stalled?

b What additional circumstances might prevent a thread's instructions from being executed in SMT mode?

c Why is a special mechanism needed for a long-executing instruction such as the interprocessor synchronisation "synch" instruction?

d The Power5's thread-throttling mechanisms are designed to ensure that multi-threading does not reduce overall throughput. Can such a reduction happen despite these measures? Explain your answer carefully.

e Suppose one thread performs poorly due to a high branch misprediction rate. Does that mean the other thread can benefit?

The five parts carry, respectively, 25%, 25%, 10%, 20%, and 20% of the marks.