1 This question concerns the Power Processing Element (PPE) of the Cell Broadband Architecture, as described in the article “Introduction to the Cell multiprocessor” (IBM Journal of Research and Development Vol.49 No.4/5 July/September 2005), which we studied during the course. A copy of this article is provided with this exam paper.

a Why does the PPE processor core occupy more silicon area (Figure 1, page 592) than an SPE (Synergistic Processing Element) processor? Identify at least three clear reasons.

b The PPE issues instruction pairs alternately from two different threads. Why might this design lead to a shorter clock cycle?

c What is the “load-use delay”? Explain how it is reduced in the PPE using delayed execution of fixed-point arithmetic.

d In Figure 2, the branch misprediction penalty is given as 23 cycles. However, 3 of these cycles are “delay” stages due to delayed execution of branch instructions. Under what circumstances does the advantage of this decision arise?

e According to the article, “a 4-KB by 2-bit branch history table with 6 bits of global history per thread is used to predict the outcome of branches”. This is somewhat ambiguous.

Draw a diagram showing the structure of a 4K-entry (6,2) *gselect* branch predictor. How many bits of the program counter address are used to produce the prediction?

f In a *gshare* predictor, the global history is combined with bits from the program counter address using an exclusive-or. Why might this be better than *gselect*?

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

2 This question partly concerns the Cell Broadband Architecture, as described in the article “Introduction to the Cell multiprocessor” (IBM Journal of Research and Development Vol.49 No.4/5 July/September 2005), which we studied during the course. A copy of this article is provided with this exam paper.

a Consider the following program fragment, in which `rand()` returns a random integer between 0 and 99:

```
for (i=0; i<100; i++) {
   int r = rand();
   if_branch1 (r > 50) { target1:  printf("pass "); }
   if_branch2 (r > 52) { target2:  printf("safely "); }
   if_branch3 (r > 54) { target3:  printf("by a mile"); }
}
```

How many branch mispredictions, on average, would you expect the PPE to incur with this code, assuming the loop has been executed beforehand, recently.

b The SPE has no branch predictor, but does have an explicit “branch hint” instruction (page 596). Using the example from part (a), show how this might be used to achieve some of the effect of a correlating branch predictor (for your convenience, the branch instruction for each `if` in the code above is labelled, `branch1`, `branch2`, `branch3`; assume they are not-taken in the “else” cases).

c The access latency for shared, off-chip memory is quite large. Accesses must be pipelined in order to take advantage of the 25.6GB/s XDR memory interface (page 597).

(i) The PPE accesses the global shared memory using ordinary load and store instructions. What limits the number of outstanding load instructions?

(ii) Suppose the PPE were a more aggressive, dynamically-scheduled design with register renaming and speculative execution. What would limit the number of outstanding load instructions?

(iii) How is memory access bandwidth maximised in the SPE design?

The three parts carry, respectively, 25%, 25%, and 50% of the marks.