Data Provided: None

The University of Sheffield

DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING

Autumn Semester 2005-2006 (2 hours)

Advanced Computer Architectures 4

Answer **THREE** questions. **No marks will be awarded for solutions to a fourth question.** Solutions will be considered in the order that they are presented in the answer book. Trial answers will be ignored if they are clearly crossed out. **The numbers given after each section of a question indicate the relative weighting of that section.**

1. a. A pipelined processor will suffer from problems arising from control-flow dependencies. Describe:

i. how these problems arise; **(3)**

ii. the methods used to overcome these problems. **(4)**

b. In some cases, a *reorder* buffer is used to solve problems caused by data/control dependencies in such processors.

i. Describe how such buffers are used and, in particular, identify the difference between instructions *completing* and *committing*. **(5)**

ii. For the following code snippet, show how a reorder buffer might be used:

```
    LOAD    @addr,R1     ; R1 ← addr
    MUL     R1,R2,R3     ; R3 ← R1 x R2
    CMP     R3,10        ; is R3 equal to 10
    JEQ     L1
    ADD     R4,R5,R5     ; R5 ← R5 + R4
    JMP     L2
L1  ADD     R4,R6,R6     ; R6 ← R6 + R5
L2  …
```

(6)

c. What is meant by *precise interrupts* and a *precise architectural state* and what is their relevance to reorder buffers? **(2)**

2. A common way to perform a division, y/R, in a floating point system, is to calculate the reciprocal of the divisor, R ($=R^{-1}$), and calculate the product $y.R^{-1}$. The reciprocal of R can be calculated using a Newton-Raphson approach by using an iterative calculation:

$$x_{n+1} = 2x_n - Rx_n^2 \qquad (1)$$

where x_0 is seeded with an *approximation* to R^{-1} and the iterative equation is calculated for increasing values of $n \rightarrow N$ until x_N converges on the value of R^{-1}.

Assume that the input number, R, is a normalised floating point number stored in the form $m \times 2^e$ where m, the mantissa, is in the range 0.5…1 and e, the exponent is an integer in the range -128…127.

To set the initial seed, a value of x_0 has to be chosen that is less than R^{-1} but is close enough to ensure that x_n converges to R^{-1} quickly. You can also assume that the number of iterations, N, required to produce an accurate value of R^{-1} can be approximated by:

$$N=4 - log_2(x_0R)$$

a. Based on the format of the numbers, show that the number of iterations required can be limited to 6. **(4)**

b. Based on the result shown in part **a.**, design a pipelined datapath block that will allow you to calculate one iteration of equation *(1)*. **(8)**

c. Show how the block in part **b.** could be used within a larger schematic to:

i. Calculate the reciprocal, R^{-1}, whilst utilising the minimum hardware. **(4)**

ii. Calculate the reciprocal, R^{-1}, whilst achieving the greatest throughput possible **(4)**

3. **a.** State Flynn's Taxonomy – giving an example for each classification. **(4)**

b. Describe the organisation of an SIMD Array Processor. Ensure that you identify what makes it SIMD. **(6)**

c. A square-connected *NxN* SIMD array holds an array of data, *f(x,y)*, distributed across it so that *f(x,y)* is held in memory location 0 of processor PE*x,y* (in the x^{th} column and y^{th} row). The array of data is to be sorted, within rows, so that the biggest datum from row *y* of data is held in PE0,*y*, the next biggest in PE1,*y*, the next in PE2,*y*, all the way to the smallest datum being held in PE*N-1*,*y*.

The PEs support the following instructions:

```
BCAST   <a>         ; output memory location <a> to N,S,E,
                      and W
INd     <a>         ; input from d (=N,S,E, or W) to memory
                      location <a>
INd     <a>,F       ; input from d (=N,S,E, or W) to memory
                      location <a> but only operate if F=1
MOVE    <a>,<b>     ; copy memory location <a> to <b>
MOVCA   <a>         ; a PE-dependent instruction that loads
                      the column address of the PE into
                      address <a>
MOVF    <a>         ; copy the lsb of the value in address
                      <a> to F
CSWP    <a>,<b>,F   ; exchange memory locations <a> and <b>
                      if the contents of <a>> contents of <b>
                      but only operate if F=1
AND     <a>,#K      ; logical and of value in address <a>
                      with constant value K
INV     F           ; Invert the F flag
```

Write a program that will sort the rows, as described above.

Hint: if each column is always performing a calculation then a program will not work but note that the CSWP instruction is conditional and the column address can be used as an operand. **(10)**

4. A set of N interconnected processors (where N is odd) is organised in a ring, as shown in **Figure 4**. Each processor is connected to its neighbouring processors by communication links that will transfer a message between the two connected processors in R_E seconds. An application, consisting of M tasks, is distributed equally across the N processors and during the course of the application each task sends T messages to every other task. If a message is sent from a processor to a non-adjacent processor it is done by store-and-forward via all of the intervening processors (using the shortest route around the ring). If a message is sent between two tasks on the same processor, the communication is assumed to take a time equal to R_I.

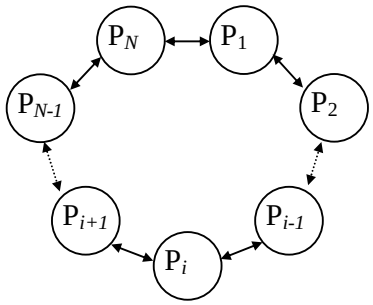

Figure 4: Processor Ring

a. Show that the total number of messages transmitted between any two processors (that is, the message originates on one processor and finishes its journey at the other processor) is:

$$2\frac{M^2}{N^2}T$$

(4)

b. From the result in **a.** and fact that messages between any two non-adjacent processors appear as individual messages across each separate communication link, show that the total number of messages sent during the execution of the application across *each* communication link is:

$$\frac{T}{4}\frac{M^2}{N^2}(N-1)(N+1)$$

(6)

c. Each task involves an internal computation time of C seconds and you may assume that internal computation is not overlapped with communication via the links. However, you may assume that all of the internal computation runs in parallel on the processors and the communication links can all operate in parallel with each other. Show that the speed-up involved in running the application on the ring of processors (relative to running the application on a single processor) is:

$$speedup = \frac{N\left(1+(M-1)\frac{TR_I}{C}\right)}{1+\frac{1}{4}\frac{TR_E}{C}\frac{M}{N}(N-1)(N+1)+\frac{TR_I}{C}\frac{(M-N)}{N}}$$

(8)

d. Show that the parallelisation of the algorithm is only worthwhile if:

$$\frac{C}{\left(\frac{R_E}{4}-R_I\right)T} > M$$

For the case where N (and, hence, M) is large (and the above is an approximation). **(2)**

NLS / NA