High Performance Microprocessor Architectures

Date:  Monday 6th June 2011
Time:  09:45 - 11:45

Please answer any THREE Questions from the FIVE questions provided

Please state clearly any additional assumptions you need to make in order to answer a question.

This is a CLOSED book examination

The use of electronic calculators is permitted provided they are not programmable and do not store text
1. Consider a classic five stage instruction pipeline modified as follows. In the execute (EX) pipeline stage, there is a choice of four parallel execution units to which an instruction may be sent: integer instructions, including all loads and stores, go to the integer unit; floating point (FP) adds and subtracts go to the FP adder unit; FP multiplys go to the FP multiplier; and FP divides go to the FP divider. The integer unit completes execution in 1 cycle and can thus accept a new instruction each cycle. The FP adder takes 4 cycles to complete, but it is pipelined in such a way that it can also accept a new instruction each cycle. The FP multiplier is similarly pipelined, but it takes 6 cycles to complete each execution. The divider is not pipelined and it takes 24 cycles to complete execution.

a) Explain the notion of a precise exception in a pipelined processor architecture.  
(2 marks)

b) Explain why it is difficult to implement precise exceptions in the multi-unit pipeline described above when instructions entering faster units are permitted to complete before those that previously entered slower units (i.e. later instructions are allowed to overtake earlier ones).  
(2 marks)

c) The following code fragment (written in C) is compiled into a short sequence of MIPS-like instructions, where \( a \) is an array of double precision FP values and \( s \) is a double precision value held in a double precision FP register.

\[
\text{for (i:=4095; i>=0; i--) a[i] := a[i] * s;}
\]

i) List an appropriate sequence of MIPS-like instructions without attempting any optimisation.  
(3 marks)

ii) Indicate where stalls will occur in the execution of a single iteration of this loop using the above pipeline. Hence or otherwise determine the number of machine cycles taken per iteration of the loop. Assume the pipeline has full forwarding for data values and state any other assumptions you need to make.  
(3 marks)

iii) Show how the instructions in the sequence might be re-scheduled (i.e. issued in a different order) by a compiler so as to reduce the number of stalls that occur. Determine the new number of machine cycles taken per iteration of the loop.  
(4 marks)

iv) Show how the compiler can unroll the loop by a factor of two and then reschedule the resulting sequence of instructions so as to achieve a further reduction in the number of stalls. State any further assumptions you need to make. Determine the new number of machine cycles taken per iteration of the loop  
(4 marks)

v) Estimate the effect of unrolling the loop by a factor of four.  
(2 marks)
2. a) Sketch the structure of a floating point (FP) execution unit that uses Tomasulo’s algorithm to implement dynamic instruction scheduling. Assume that the unit has a pipelined FP adder and a pipelined FP multiplier which operate in parallel and can both accept a new operation each cycle. Assume also that the unit has load and store buffers to handle memory accesses.

(4 marks)

b) Describe how instructions are dynamically scheduled in the above structure. Your description should concentrate on the way in which the correct data is assembled from registers, the execution units, or the memory. You should also explain how the correct data ends up in the registers or gets written to memory, and how read-after-write hazards are resolved for memory accesses.

(6 marks)

c) Explain why it is important that branch behaviour should be predicted and speculated upon in such a dynamic instruction scheduling system.

(1 mark)

d) The following loop of MIPS-like instructions is executed in a speculative dynamic instruction scheduling system like that described by your answers to the above parts and which executes branches in deferred fashion. Assume that a new instruction can be issued every cycle and that the branch predictor always correctly predicts that the branch will be taken.

```
start:   LD     f0, 0(r1)
ADD   f4, f0, f2
SUBI   r1, r1, #8
BNEZ   r1, start
SD     f4, 8(r1)
```

i) Draw a timing diagram that shows how the execution of successive iterations of the loop soon settles into a pattern in which a new instruction is issued and an old instruction effectively completes every machine cycle. State clearly any additional assumptions you need to make.

(6 marks)

ii) What is the minimum number of reservation stations (including load and store buffer entries) that is necessary to sustain execution at the above rate? Explain your reasoning.

(3 marks)
3. a) A *superscalar* microprocessor attempts to issue more than one instruction per machine cycle.

i) Why is this important for a high performance microprocessor? (1 mark)

ii) What choices are available for issuing instructions into a superscalar architecture and what are the benefits and disadvantages of each choice? (4 marks)

b) The following source code extract (written in C) implements a dense square matrix multiplication, where a, b and c are $n \times n$ arrays of double precision floating point values.

```c
for (i:=0; i<n; i++)
    for (j:=0; j<n; j++)
        for (k:=0; k<n; k++)
            c[i,j] += a[i,k] + b[k,j];
```

A compiler generates the following sequence of MIPS-like instructions for the innermost loop of the source code extract under the assumptions stated in the comments below.

```
start:   LD     f0, 0(r2)     // r2 points to a[i,k] ..
         // .. starts at a[i,0]
LD     f2, 0(r3)     // r3 points to b[k,j] ..
         // .. starts at b[0,j]
MULD   f6, f0, f2
ADDDD  f4, f4, f6    // f4/5 contains c[i,j] ..
         // .. starts at 0.0d
SUBI   r1, r1, #1    // r1 starts at n
ADDI   r2, r2, #8    // finds next a[i,k]
BNEZ   r1, start     // exits when r1 reaches 0
ADDI   r3, r3, r5    // r5 holds 8*n so as to ..
         // .. find next b[k,j]
```

This loop is executed on a dynamically scheduled, speculative 2-way superscalar microprocessor which has one pipelined integer unit alongside one pipelined floating point unit and which executes branches in deferred fashion. Loads, stores and branches are executed in the integer unit. A maximum of one integer instruction plus one floating point instruction may be issued in each machine cycle. You may assume that the branch is always correctly predicted to be taken.
Question 3 continues from the previous page

i) What is the minimum number of machine cycles needed to issue (i.e. commence execution of) each iteration of this loop in the form shown? State any assumptions you make about how the hardware helps to achieve this rate of instruction issue. (2 marks)

ii) Could compiler re-scheduling of the instruction sequence reduce the number of machine cycles needed to issue each iteration? If so, what would be a suitable new sequence and what is the corresponding number of machine cycles required? (3 marks)

iii) Show how a compiler can unroll this loop by a factor of two, and re-schedule the new sequence of instructions to find the best 2-way instruction schedule per iteration of the resulting loop and the corresponding number of cycles to issue each iteration. What property of the original code fragment does this loop unrolling rely on? (4 marks)

iv) Repeat part c) iii), but using a loop unrolling factor of eight. What do you conclude about the potential benefits of loop unrolling? (4 marks)

v) Explain in outline how the compiler might deal with unrolling of loops for which the total number of iterations is not known at compile-time. (2 marks)
4. a) Outline the principles of *value prediction* and explain how it can lead to improved processor performance. You do not need to give details of the hardware structure of a processor using the technique. (3 marks)

b) Given a sequence of integer values of the general form: \(x(0), x(1), x(2), x(3), \ldots x(i), x(i+1), \ldots\), where \(x(i+1) = x(i) + k*i\) and \(k\) is a constant, show how it is possible to predict the next value in the sequence from the last three values produced. What is predicted when fewer than three prior values have been produced? (4 marks)

c) An integer instruction in a program produces the values 0, 0, 1, 3, 6, 10, 15, 21, 28, 0, 0, 1, 3, 6, 10, 15, 21, 28, 0, 0, 1, 3, 6, 10, \ldots in a repeating sequence. Explain how the value predictor from part b) will react to this sequence. (4 marks)

d) A *context value predictor* is proposed which maintains a partial history of prior values by storing the least significant 2 bits of the values produced by the instruction on its two immediately prior executions (you may assume that this history for this instruction is captured in full). Sketch the storage structure required to implement this new predictor. Explain how the contents of this storage structure are manipulated as the context value predictor reacts to the repeating sequence given in part c). Compare the resulting behaviour of the context value predictor with that described in your answer to part c). (6 marks)

e) Explain how you would modify the above context value predictor so that it can correctly predict all values in the given repeating sequence, once the prior history table has been appropriately filled. (3 marks)
5. a) Describe the techniques of direct mapped, set associative and fully associative caching, comparing the potential performance and cost of each. (6 marks)

b) Explain what cache pre-fetching is and how it may be implemented. (2 marks)

c) A processor has a 16k byte 2-way set associative level 1 data cache using 16 byte cache lines. The memory is byte addressed and the cache line index is obtained by masking the memory address with 0x00001FF0 and shifting the result right 4 places. The following program fragment (written in C) is compiled and executed.

```c
for (i:=0; i<65535; i++) s += a[i] + b[i] + c[i];
```

Assume that the loop index i and the value of s are both held in registers, and that the three arrays (a, b and c) each comprise 64k words of 32-bit integer values which are stored consecutively in memory, starting at address 0x10000000 (for element a[0]).

i) What is the minimum possible number of cache misses that must occur in order for this code fragment to execute? What is the corresponding cache hit rate? Explain your reasoning. (2 marks)

ii) How many cache misses will occur if the cache uses a Least Recently Used (LRU) replacement policy? What is the corresponding cache hit rate? Explain your reasoning. (4 marks)

iii) What would be the effect of changing the cache so that it uses a Random replacement policy? (2 marks)

iv) What would be the effect of adding a single entry victim cache to the system, using either of the above replacement policies? Explain your reasoning. (4 marks)

END OF EXAMINATION