Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

M.Sc. in Mathematics & Computational Science

Fundamentals of High Performance Execution

Tuesday 20th January 2009

Time: 09:45 – 11:45

Please answer TWO Questions from the FOUR questions provided

This is an OPEN book examination

The use of electronic calculators is permitted provided they are not programmable and do not store text
Please answer TWO Questions from the FOUR questions provided

1. a) Describe the format and operation of the main classes of instruction that are used in a typical computer with a register-based load/store instruction set architecture (ISA). Include details of any memory accesses made during execution of each class of instruction. (5 marks)

b) Explain the basic principles of pipelined execution of instructions. Discuss the gains that can be made and the costs that have to be paid in implementing an instruction pipeline, taking into account the need to minimise the effects of hazards. (6 marks)

c) The following fragment of C code is compiled without optimisation into MIPS-like assembler code.

```c
for (i = 4096; i>0; i--) {
    x[i] = x[i] + s;
}
```

The resulting assembler code is shown below.

```
loop: LD F0, 0(R1)
ADDD F4, F0, F2
SD 0(R1), F4
SUBI R1, R1, #8
BNEZ R1, loop
NOP
```

This code is executed on a 5-stage instruction pipeline in which there is (1) a one cycle stall if a load is followed by an immediate use of the value loaded, (2) the effect of a branch instruction is taken one instruction after the branch is issued, and (3) the floating point add operation takes a total of 3 cycles to execute in the EX stage of the pipeline.

i) Calculate the total number of clock cycles that are required to execute a single iteration of the given loop. (3 marks)

ii) Suggest a different order of execution for (possibly modified versions of) these instructions that will execute faster. Calculate the resulting number of clock cycles now needed to execute a single iteration of the loop. (4 marks)

iii) What other technique or techniques can be applied to this loop to further decrease the number of clock cycles needed to execute it? (2 marks)
2. a) Explain the distinction between a Finite State Machine (FSM) and a Turing machine. Include a discussion of the different computing capabilities of these two primitive computers. (6 marks)

b) The address of a memory location in a conventional (register-based) computer architecture is limited to the values that can be represented in a single word (of the prevailing wordlength). How many distinct locations can be addressed in a computer with a wordlength of 32-bits? (1 mark)

c) In what ways can the finite memory described in part b) be utilised and extended in such a way as to mimic the behaviour of the infinite tape memory associated with a Turing Machine? (5 marks)

d) The following C program computes the factorial of 5 using a bilinear recursive algorithm.

```c
#define size 5

int fac (int n,m) {
    int mid = (n+m)/2; /* computes the floor/*
    if(n==m)
        return n
    else
        return fac(n, mid)*fac(mid+1, m);
}

int main(){
    int answer;
    answer = fac (1, size);
    /* assume answer printed here/*
}
```

Show how activation frames, loaded and unloaded to and from the stack, are used during the execution of this program to complete the final answer. (8 marks)
3.  
   a) Explain the basic principles of cache memory as a means for providing fast access to memory locations in a large (and therefore essentially slow) address space. Include a discussion of the options available for: (1) placement of a cache line when it is first brought into the cache memory; (2) writing a changed cache line back to the main memory; and (3) displacement of a cache line when a new cache line is needed but there is no vacant place to put it in the cache memory. Describe the conditions under which a cache memory will not deliver fast access to memory.  
   (9 marks)

   b) A three-level cache hierarchy contains components with the characteristics shown in the following table.

<table>
<thead>
<tr>
<th></th>
<th>Level-1 cache:</th>
<th>Level-2 cache:</th>
<th>Level-3 cache:</th>
<th>Main memory:</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>32 kbytes</td>
<td>512 kbytes</td>
<td>8 Mbytes</td>
<td>16 Gbytes</td>
</tr>
<tr>
<td>local access time</td>
<td>1 ns</td>
<td>5 ns</td>
<td>20 ns</td>
<td>100 ns</td>
</tr>
<tr>
<td>expected hit rate</td>
<td>97%</td>
<td>85%</td>
<td>60%</td>
<td></td>
</tr>
</tbody>
</table>

Notes: 1 ns = $10^{-9}$ seconds.

The expected hit rates are all local rates, that is, the percentage of accesses referred to the associated level of the hierarchy by a miss in the previous level that result in a hit.

Calculate the average access time for this memory system.  
(5 marks)
(Question 3 continues from the following page)

c) A 4 Gbyte executable image (i.e. using the full range of byte addresses available in a computer with a wordlength of 32-bits) can be mapped into a physical memory of a larger (or smaller) size using the virtual memory scheme known as paging. Assuming a physical memory size of 16 Gbytes and a page size of 64 kbytes, answer the following.

i) Using an appropriate diagram, explain how the 32-bit virtual address is translated into a 34-bit physical address. (2 marks)

ii) How many bits of the virtual address are used to define the offset within the specified page? (1 mark)

iii) How many pages does the largest possible executable image contain? (1 mark)

iv) How much memory is required to hold the largest possible page table for an executable image? (2 marks)
4. The following grammar and associated semantic rules describe arithmetic expressions involving identifiers, \texttt{id}, integer constants, \texttt{num}, and the operators +,−,×,÷ (note that this is not our conventional understanding of the evaluation of arithmetic expressions):

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantic Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{expr} → \texttt{expr} + \texttt{factor}</td>
<td>\texttt{expr.t} \triangleright= \texttt{expr.t} \parallel \texttt{factor.t}</td>
</tr>
<tr>
<td>\texttt{expr} → \texttt{expr} − \texttt{factor}</td>
<td>\texttt{expr.t} \triangleright= \texttt{expr.t} \parallel \texttt{factor.t}</td>
</tr>
<tr>
<td>\texttt{expr} → \texttt{expr} \times \texttt{factor}</td>
<td>\texttt{expr.t} \triangleright= \texttt{expr.t} \parallel \texttt{factor.t}</td>
</tr>
<tr>
<td>\texttt{expr} → \texttt{expr} ÷ \texttt{factor}</td>
<td>\texttt{expr.t} \triangleright= \texttt{expr.t} \parallel \texttt{factor.t}</td>
</tr>
<tr>
<td>\texttt{expr} → \texttt{factor}</td>
<td>\texttt{expr.t} \triangleright= \texttt{factor.t}</td>
</tr>
<tr>
<td>\texttt{factor} → (\texttt{expr})</td>
<td>\texttt{factor.t} \triangleright= \texttt{expr.t}</td>
</tr>
<tr>
<td>\texttt{factor} → \texttt{id}</td>
<td>\texttt{factor.t} \triangleright= \texttt{'id'}</td>
</tr>
<tr>
<td>\texttt{factor} → \texttt{num}</td>
<td>\texttt{factor.t} \triangleright= \texttt{'num'}</td>
</tr>
</tbody>
</table>

a) By constructing the annotated syntax trees, show that

\[ a \times b + c - 2 \div d - a \]

where \(a\), \(b\), \(c\), \(d\) are identifiers, and \(2\) is an integer constant, is a valid arithmetic expression and obtain the postfix notation for the expression. \(6\) marks

b) Rewrite expressions, using brackets only where necessary, so that the evaluation respects our conventional understanding that \(\times, \div\) have higher precedence than +,−. \(4\) marks

c) Extend the grammar (and the associated semantic rules) so that it respects our conventional understanding of precedence of operators, without the need to bracket operations, and construct the associated annotated syntax tree for the expression given in part a). (Hint: you will need to introduce an additional nonterminal symbol \texttt{term}.) \(6\) marks
d) Given an abstract stack machine with the following instructions:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>push v</td>
<td>push integer constant v onto the stack</td>
</tr>
<tr>
<td>rvalue l</td>
<td>push contents of data location l onto stack</td>
</tr>
<tr>
<td>lvalue l</td>
<td>push address of data location l onto stack</td>
</tr>
<tr>
<td>+</td>
<td>Pop and add two topmost values on stack and push result onto stack.</td>
</tr>
<tr>
<td>−</td>
<td>Pop the two topmost values from stack, subtract the topmost value from the second topmost value and push result onto stack.</td>
</tr>
<tr>
<td>×</td>
<td>Pop and multiply two topmost values on stack and push result onto stack.</td>
</tr>
<tr>
<td>÷</td>
<td>Pop the two topmost values from stack, divide the second topmost value by the topmost value and push result onto stack.</td>
</tr>
<tr>
<td>:=</td>
<td>the r-value on top of the stack is placed in the l-value below and both are popped.</td>
</tr>
</tbody>
</table>

Write machine instructions to perform the assignment

\[ id := a \times b + c - 2 \div d - a \]

where the expression is specified by the grammar of part c). (4 marks)