One and a half hours

"ARM Instruction Set Summary" is attached

Question ONE is COMPULSORY

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Fundamentals of Computer Architecture

Date: Friday 24th January 2014
Time: 14:00 - 15:30

Please answer Question ONE and ONE other Question

Use a SEPARATE answerbook for each SECTION

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
Section A

1.

a) Explain briefly the following instruction styles. (3 marks)
   i) 3-address
   ii) 1-address
   iii) load-store

b) What is a stack? How does it work? Why is it needed in modern computing systems? (3 marks)

c) Perform the following base translations. Do not use binary as an intermediate representation. (3 marks)
   i) 0b101101011010 to Hexadecimal
   ii) @3121 to Binary
   iii) 411 to Binary
   iv) 0x357 to Dec
   v) 0b1100110010 to Dec
   vi) 349 to Oct

d) How many bits would we need to represent: (1 mark)
   i) The 256 colours supported by the VGA Mode 13h standard.
   ii) The 3,120,000 processing cores of the largest supercomputer in the Top500 list of supercomputers in June 2013.

e) During your lectures on interrupt vectors a statement was made:
   “In order for the processor to execute interrupts, a mapping must exist between interrupt and handler.”
   i) What mechanism is used to achieve this mapping? (1 mark)
   ii) Briefly explain this mechanism? (1 mark)
f) The table, in figure 1.f, shows a code snippet from a very simple interrupt vector handler. Copy the table into your answer book and then describe in full what happens when the ARM program is obeyed.
In the table clearly describe the movement of information (both numbers and instructions) between memory, registers and the CPU in the comments column, and how the values in the registers R0, R1, and memory change, at each step. (4 marks)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operands</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR</td>
<td>R1, Acknowledge</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>R0, #1</td>
<td></td>
</tr>
<tr>
<td>STR</td>
<td>R0, [R1]</td>
<td></td>
</tr>
</tbody>
</table>

Question figure 1.f. A table depicting part of a very simple interrupt handler.

g) Explain base plus index addressing as implemented in the ARM processor. (4 marks)
Section B

2.

a) Name the four steps and very briefly describe each step of an assembler.
   (4 marks)

b) Given that two steps in the assembly process – steps two and three – are aligned to structure and meaning.
   
   i) Compare and contrast the two by giving a detailed description of steps two and three of an assembler – highlighting the differences in your answer book using the template in figure 2.b.; whilst also naming the assembly steps, again – in the first column.
   (4 marks)

<table>
<thead>
<tr>
<th>Step</th>
<th>Detailed describe steps 2 &amp; 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2:</td>
<td></td>
</tr>
<tr>
<td>Step 3:</td>
<td></td>
</tr>
</tbody>
</table>

   Question figure 2.b. A table; showing detailed describe steps 2 & 3.

   ii) A compiler has five steps; name and briefly describe the compiler’s last step; give examples of how it archives the fifth step’s goal. (3 marks)

   iii) State the three [hierarchical] abstraction levels of programming languages; also state which is human and which machine understandable. (3 marks)

c) What is bytecode in the context of Java and how does it get executed by a real processor? (3 marks)

d) Name the three Operating Systems kernel managers and briefly describe each. (3 marks)
Section C

3.

a) Explain briefly the difference between Instructions, Pseudo-instructions and Directives and give an example of each. (3 marks)

b) The following code selects an output text depending on the value of three positive integer variables (a, b and c). The different output texts are stored in a table with fixed-size elements of 64 bytes. Transform the code into ARM assembly simplifying and optimising it as much as possible. Explain your optimisations. Assume the inputs are always valid, i.e. a, b and c are lower than the number of elements in the table. (5 marks)

```
if (a > b && a>c)
    result=table[a]; // Address of the first character of element a
if (b > a && b>c)
    result=table[b]; // Address of the first character of element b
else
    result=table[c]; // Address of the first character of element c
```

c) The Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, ...) has a number of applications in diverse knowledge areas (mathematics, computer science and biology, to cite just a few). The following code presents an iterative algorithm to compute the $n^{th}$ Fibonacci number ($n \geq 0$) into variable f. Transform it into ARM assembly simplifying and optimising it as much as possible. Explain your optimisations. (6 marks)

```
if (n==0)
    f=0; // Fibonacci(0)=0
else if (n==1)
    f=1; // Fibonacci(1)=1
else {
    f1=1; // Fibonacci (i-1)
    f2=0; // Fibonacci (i-2)
    for (i=2; i<=n; i++) {
        f=f1+f2; // Fibonacci(i)=Fibonacci(i-1)+Fibonacci(i-2)
        f2=f1;
        f1=f;
    }
    f
}
```
d) The following function computes the $n^{th}$ Fibonacci number recursively. Transform it into ARM assembly simplifying and optimising it as much as possible. Explain your optimisations and how your code passes parameters and results. (6 marks)

```c
int fib (int n){
    if (n==0)
        return 0; // Fibonacci(0)=0
    if (n==1)
        return 1; // Fibonacci(1)=1
    else
        return fib(n-1)+fib(n-2);
}
```