One and a half hours

Closed Book Examination
(A copy of an "ARM Instruction Set Summary" is attached)

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Fundamentals of Computer Architecture

Thursday 22\textsuperscript{nd} January 2016?

Time: 14:00 – 15:30?

Please answer Questions A1 and B1
and
ONE other Question: either A2 or B2

\textbf{Use a SEPARATE answer book for each SECTION}

For full marks your answers should be CONCISE as well as ACCURATE. Marks will be awarded for reasoning and method as well as being correct. Excessively verbose answers will be penalised.

The use of electronic calculators is NOT permitted.
Section A

A1. **Computer Architecture** [Javier Navaridas-Palma]

a) Translate the decimal number $347_{10}$ to hexadecimal and the octal number $135_{8}$ to decimal **without using binary** as an intermediate representation. (4 marks)

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>347</td>
<td>1B</td>
<td>135</td>
<td>15B</td>
</tr>
</tbody>
</table>

Practical - 2 marks each. 1 mark if binary is used or no process is shown

$$347_{10} \text{ to Hex}$$

347 | 16 | 11 → B
21 | 16 | 5 → 5B
1 | 16 | 1 → 15B

$$135_{8} \text{ to Dec}$$

$$1*8^2 + 3*8^1 + 5*8^0 = 64 + 24 + 5 = 93$$

b) Explain **briefly** the following addressing modes and their pros and cons. (3 marks)

i) Direct addressing
   - **Pros:** simple to understand and implement
   - **Cons:** not very flexible, needs to know every address in advance. Instruction needs to be larger than 1 word.

ii) Base+Offset addressing
   - **Pros:** flexibility, allows to move around the target address without changing it.
   - **Cons:** extra instructions may be needed to calculate displacement

iii) Post-index addressing
   - Similar to base+offset addressing but updates the value of the base register after accessing memory.
   - **Pros:** flexibility, changing the target address saves an arithmetic instruction.
   - **Cons:** less common address mode, requires space in instruction space.

Bookwork – 0.5 mark for covering the points below up to 3 marks.

**Direct addressing:** the target memory address is contained within the instruction typically used for constants

- **Pros:** simple to understand and implement
- **Cons:** not very flexible, needs to know every address in advance. Instruction needs to be larger than 1 word.

**Base+Offset addressing:** the target address is calculated from a base address held elsewhere (typically a register) and a displacement (either in a literal or a register).

- **Pros:** flexibility, allows to move around the target address without changing it.
- **Cons:** extra instructions may be needed to calculate displacement

**Post-index addressing:** Similar to base+offset addressing but updates the value of the base register after accessing memory.

- **Pros:** flexibility, changing the target address saves an arithmetic instruction.
- **Cons:** less common address mode, requires space in instruction space.
c) Explain the differences between 3-address style instructions, 1-address style instruction and load-store style instructions.  

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3-address</strong></td>
<td>All operands and results are stored in memory. Implies many memory operations per instruction which means instructions are executed very slowly.</td>
</tr>
<tr>
<td><strong>load-store</strong></td>
<td>All operands and results are stored in registers. Memory accesses need to be explicitly written by the programmer, which means more instructions are needed.</td>
</tr>
<tr>
<td><strong>1-address</strong></td>
<td>A tradeoff between the two styles above. Most operations are done using registers, but one of the operands or the result can be directly read from or stored in memory.</td>
</tr>
</tbody>
</table>

Bookwork - 1 mark each.
A2. [Javier Navaridas-Palma]

a) Explain the similarities and differences between the following pairs of ARM mnemonics:

(4 marks)

i) STR and STMFD
ii) SUB and SUBS
iii) ADD and ADDEQ
iv) ADR and ADRL

Bookwork - 1 mark each.

**STR and STMFD:** Both of them store data in memory. The first stores from a single register to a memory location, whereas the second can store several registers to consecutive memory locations. STR can have more elaborated addressing modes (e.g. indexing).

**SUB and SUBS:** Both of them will perform a subtraction but the second one will modify the status register so that the following conditional instructions depend on its result.

**ADD and ADDEQ:** Both of them perform an addition. The first one will always be executed, whereas the second one is only executed if condition EQ (equal) is true (from last CMP or …S instruction).

**ADR and ADRL:** Both of them are *pseudo instructions* that are used to put a memory address in a register. ADR will be translated into a single instruction and so has a limited range. ADRL can be translated into many instructions and can reach any memory location.
b) Consider the following switch statement to manage a keyboard-controlled menu in an embedded system:

```c
switch (option) {
    case 'd': delete();
        break;
    case 'm': move();
        break;
    case 'r': rename();
        break;
    default: invalid_option();
        break;
}
```

Describe, without giving any code, two different ways of implementing it in ARM. Which one of them would you choose if reducing memory footprint was the main concern of the target embedded system: (6 marks)

<table>
<thead>
<tr>
<th>Application (Critique) – 2 mark per implementation + 2 marks for efficiency discussion.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 possible implementations:</td>
</tr>
<tr>
<td>- Nested IFs</td>
</tr>
<tr>
<td>- An address table</td>
</tr>
</tbody>
</table>

Discussions on smart or exotic implementations may get some extra marks if justified adequately.

Efficiency:
For a switch with only 3 cases as the one in the example, using nested IFs is a better implementation because, at most, 3 conditions need to be checked (i.e. 6 instructions), whereas in a table-based implementation, we would need a large table containing lots of defaults (2 instructions + 15 elements in the table).
c) You need to implement an error printing method which, given an error code as a parameter (in R0) it looks up in a table for the corresponding printing colour and error string. This table (labelled `error_table`) has 20 elements with a fixed width of 16 bytes. In each element the least significant byte is used to store a colour code and the other 15 bytes store an error string (see below). In order to print the error code in the required colour in a terminal you need to use the method `print_formatted` which accepts 2 parameters: the colour to be used in R0 and the address of the first character to print in R1. Your function needs to return a completion code in R0: 1 if the printing was successful or -1 if there was any error.

```
<table>
<thead>
<tr>
<th>Colour</th>
<th>Error string</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```

Make sure your code is safe and try to optimise it as much as you can, explaining what optimizations you do. (10 marks)

Problem solving – similar to the table look-up we saw in lecture 11
2 marks for checking the value of the parameter is in range + 1 mark for optimizing with HS
```
print_error  CMP r0,#0           \                  CMP r0,#20
            MOVLT r0,#0          \->    MOVHS r0,#-1
            CMP r0,#20           /                  MOVHS pc,lr
            MOVGE r0,#0         /                  MOVGE pc,lr
            MOVGE pc,lr        /
```

2 marks for computing addresses correctly + 1 mark for optimizing with LSL (or +0.5 for MLA) + 1 mark for optimizing with post-indexing
```
ADRL r1,error_table

            MOV r2,#16          \                  ADD r1,r1,r0 LSL #4
            MUL r0,r0,r2       \->                  ADD r1,r1,r0
            ADD r1,r1,r0       /

            LDRB r0,[r1]        \                  LDRB r0,[r1],#1
            ADD r1,r1,#1       \->
```

2 marks for safe-keeping LR into the stack + 1 mark for calling the method properly
```
PUSH {LR}
BL print_formatted
MOV r0,#1
POP {PC}
```
Section B

B.1 [Richard Neville]

a) With respect to the ‘stack’ in a Java Virtual Machine? Brief explain the types of information the ‘stack’ holds. (2 mark)

1.e.

Bookwork (2 marks):

The following points should be covered to some degree in the answer:

The stack memory is basically used for:

Parameter passing;
Saving registers; and
Temporary variables.

In answer to:-
Brief explain the types of information the ‘stack’ holds:

To [hold] parameters passed to methods;
[Saves] Local variables;
[Saves] Temporary variables;
[Saved] registers;
[Saves] Return links.

2 marks for an answer that depicts all the salient facts in a sensible way; all facts are correctly delineated and briefly described;
1 ½ marks for a right-lines approach;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): Lecture 18: Java Memory Usage.

TOTAL marks (2 marks) [2]
b) With respect to the JVM and in relation to the “3-steps taken to create a new object” briefly explain each step; draw up a set of 3-diagrams (as outlined in figure 1.e.), fully label and annotate each, given this Java code:

```java
String s;
s = new String("abc");
```
and the address it is allocated is 1000. (4 marks)

**Application (Critique) (4 marks):**

The following points should be covered to some degree in the answer:

3-Steps taken when “we create a new object:”

- **Step 1:** A new section of memory is obtained (e.g. at address 1000);
  
  which is the Right size for [the] object;

- **Step 2:** Object is initialised; and [finally]

- **Step 3:** Address of object is written to ‘s.’

![Diagrams](image)

**4 marks** for an answer that depicts all the salient facts in a sensible way; good briefly described and expression is evaluated totally correct;

**3 marks** for correct answer but not detailed enough;

**2 mark** for a right-lines approach;

**1 mark** for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): Lecture 18: Java Memory Usage.

TOTAL marks (4 marks) [6]
Operating System Kernel consists of basically five main functions; name the five main functions a Kernel performs for an (OS) and briefly describe each.

The following points should be covered explicitly in this exam; to gain full marks for your answer:

Five of the main functions a Kernel performs for an (OS) are:

1. File Manager – manages files on disk and the file structure;
2. Device Drivers – control I/O devices;
3. Memory Manager – allocates memory to programs;
4. Scheduler and Dispatcher – decides which program to run and ensures that it has the correct resources; and
5. Network manager – controls networking (connections to other computers)...

4 marks for an answer that depicts all the salient facts in a sensible way; good briefly described and expression is evaluated totally correct;
3 marks for correct answer but not detailed [enough];
2 mark for a right-lines approach;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): Lecture 15: System Software.

TOTAL marks (4 marks) [10]
B2. [Richard Neville]

a) In your lectures you were introduced to code, in the form of, a loop for ‘polling a simple device;’ which would be something like that presented in figure 3.a. State [explicitly] the function each line performs in the comment column; when you copy figure 3.a into your answer book.

Hint: this means you should be as: concise and explicit in the explanation of the: labels, instructions [operation], operand(s), registers, variables etc.; as possible.

(4 marks)

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Operand</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop</td>
<td>ADR</td>
<td>R1, status_reg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDR</td>
<td>R0, [R1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TST</td>
<td>R0, #0x80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BEQ</td>
<td>loop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADR</td>
<td>R1, data_reg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDR</td>
<td>R0, [R1]</td>
<td></td>
</tr>
</tbody>
</table>

figure 3.a., Question figure 3.a., A table; showing out-of-order code for a loop that can check a table of peripherals.

3.a.
Application (4 marks):
The following points should be covered to some degree in the answer:

The typical commented code is:

loop   ADR   R1, Status_Reg ; R1 points to status register.
       LDRB  R0,[R1]    ; read status, R0 contains value of status register.
       TST   R0, #0x80  ; test ready bit (bit 7) of status register.
       BEQ   loop       ; if not ready, try again; until bit (bit 7) SET.
       ADR   R1, Data_Reg ; R1 points to data register.
       LDRB  R0,[R1]    ; ready, [so] read data register [content].

4 marks for majority of above;
3 marks for correct answer but not [enough],
2 mark for a right-lines approach,
1 marks for some basic understanding (or attempt).
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): Lecture 14: Input/Output (2).

TOTAL marks (4 marks) [4]
b) In the context of data exchange between CPU and peripherals. Differentiate between the two main data exchange protocols; polling and interrupts (their different steps/phases).

3.b. Application (example differentiate) (4 marks):
The following points should be covered to some degree in the answer:

1. Polling basically has two phases:
   Polling phase 1: CPU polls the status register to check (e.g.) if a key has been pressed.
   Polling phase 2: Then it reads the data register [in the peripheral] in order to transfer the data from the peripheral to the CPU. [Note this is in the context of a keyboard connected to a peripheral.]

2. Interrupts nominally involve three steps:
   Interrupts step 1: When the CPU starts the interrupt handler it first checks the status register. If the status [specified by a set bit] is incorrect the handler initiates an error handler.
   Interrupts step 2: As long as the status facilitates a data transfer; the data register is then read; and the data is saved [in the appropriate location].
   Interrupts step 3: Finally an acknowledgement is written back to the peripheral.

4 marks for majority of above; e.g. for an answer that orders all lines correctly (2 marks for a detailed and concise description.),
3 marks for correct answer but not concise [enough],
2 marks for some information & for a right-lines approach,
1 mark for some basic understanding (or attempt).
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): Lecture 14: Input/Output (2).

TOTAL marks (4 marks) [8]
c) Explain what is meant by the term Java Bytecode? (4 marks)

3.c. Application (example) (Critique) (4 marks):
The following points should be covered to some degree in the answer:

1. The Java compiler does not produce real machine instructions. It produces instead, bytecode, which is a stack based (zero address) instruction set.
2. This has the advantages of portability and compactness.
3. The simplest way to execute bytecode is with a software interpreter.
4. However, performance considerations have led to the development of dynamic compilation virtual machines.
5. Here bytecode is initially interpreted but, if monitoring shows that the code is being heavily executed, the bytecode is translated to native machine code.
6. Plus any and all [related] full explanation(s) [diagrammatic and textual] – re. diagram – such as those depicted in lectures, entitled: Java Bytecode; Why Java Bytecode? Zero Address Instructions; etc…

(4 mark)

4 marks for an answer that depicts all the salient facts in a sensible way; all keywords correctly delineated [in context] and aligned to appropriate step,
3 marks for correct answer but not all keywords aligned correctly,
2 mark for a right-lines approach; but more alignment errors,
1 mark for some basic understanding (or attempt).
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lecture 14: Input/Output (2).
TOTAL marks (4 marks) [12]
d) Explain how an expression of the form \( x = (a-b) \times (c+d) \) is evaluated using [a sequence of] bytecode instructions. Typical bytecode instructions are: PUSH and POP.

3.d. Application (example details) (4 marks):
The following points should be covered to some degree in the answer:

Bytecode instructions are zero address, that is they usually don’t specify the source and destination of their operands. Instead these are implicit as the top locations of the stack (the exact use depends on the instruction). PUSH and POP instructions move data between permanent memory and the stack.

The expression \( x = (a-b) \times (c+d) \) is evaluated thus:

\[
\begin{align*}
\text{PUSH} & \ a \\
\text{PUSH} & \ b \\
\text{SUB} & \ \\
\text{PUSH} & \ c \\
\text{PUSH} & \ d \\
\text{ADD} & \ \\
\text{MUL} & \ \\
\text{POP} & \ x \\
\end{align*}
\]

4 marks for majority of above; e.g. for an answer that orders all lines correctly (2 marks for a detailed and concise description.),
3 marks for correct answer but not concise [enough],
2 mark for some information & for a right-lines approach,
1 marks for some basic understanding (or attempt).
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): Lecture 17: Input/Output (2).

TOTAL marks (4 marks) [16]
e) In general array access involves a computed index which is added to the base of the array. Draw up a diagram that depicts ‘a’ the base, ‘i’ the index. The array has ten elements of 32 bits each. Depict the case where ‘i=5’ in your diagram; diagram should be fully labelled and annotated. (4 marks)

3.e. Application [iii. A OxCamb. type final Q – how much do they really know?] (4 marks):

Application of their knowledge via diagrammatic depiction.

The following points should be covered to some degree in the answer:

```

Diagram should be fully labelled and annotated.

4 marks for an answer that depicts all the salient facts in a sensible way; all keywords correctly delineated [in context] and aligned to appropriate step,
3 marks for correct answer but not all keywords aligned correctly,
2 mark for a right-lines approach; but more alignment errors,
1 mark for some basic understanding (or attempt).
Reference Learning Resources, Background Reading, and Lecture itself for detailed information;
Lecture(s) No(s.): Lecture 19:Arrays (1): Array Implementation.
```