Special instructions, e.g.: On-Line Examination
This paper will be taken on-line and this is the paper format which will be available as a back-up.

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Fundamentals of Computer Architecture

Thursday 22nd January 2012?

Time: 14:00 – 16:00?

Please answer any THREE Questions from the FOUR questions provided?

Use a SEPARATE answerbook 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

The use of electronic calculators is permitted provided they are not programmable and do not store text.
Section A

1. **Computer Architecture**

e) Explain the function of an assembler in the generation of binary code that can be executed by the ARM processor. (2 marks)

<table>
<thead>
<tr>
<th>Bookwork (4 marks):</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following points should be covered to some degree in the answer:</td>
</tr>
<tr>
<td>A processor executes instructions which are represented as binary patterns in the memory. However, it is very difficult to write programs directly in binary patterns and so a textual form, assembly language, is used. An assembler translates this textual form into the binary form.</td>
</tr>
<tr>
<td>OR</td>
</tr>
<tr>
<td>1) Lexical (word) analysis;</td>
</tr>
<tr>
<td>The process of converting a sequence of characters into a sequence of tokens…</td>
</tr>
<tr>
<td>2) Syntactic (structure) analysis;</td>
</tr>
<tr>
<td>Checking instructions are legal…</td>
</tr>
<tr>
<td>3) Semantic (meaning) analysis;</td>
</tr>
<tr>
<td>Check user-defined names: declared exactly once…</td>
</tr>
<tr>
<td>4) Code generation.</td>
</tr>
<tr>
<td>Translate to binary machine code…</td>
</tr>
</tbody>
</table>

2 marks for an answer that depicts all the salient facts in a sensible way; all three managers correctly delineated and briefly described; 2 ½ marks for correct answer but not detailed [enough]; 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 16: Assemblers and Compilers.

TOTAL marks (2 marks) [2]

**Marker’s feedback 1.e.**

**Pedagogic assessment [criterion]:**
The question assesses lecture 16 learning objective 2, 3 and 4 indirectly; 2. Differentiate between an Assembler and a Compiler; 3. List the four Assembly Steps; and 4. Explain in detail what each Assembly Step does.
Well done, most of you were able to state what function an assembler performs in the generation of binary code that can be executed by the ARM processor.
The question’s answer should clearly evidence knowledge of required salient facts relating to the function of an assembler in the generation of binary code that can be executed by the ARM processor.
In the answer [some of] the following terminology (keywords and naming conventions) should be utilised in context; for example: processor, executes, instructions, binary, patterns, memory, textual, assembly, language, translates. Main differentiation that must be clearly evidenced in your answer is that: the assembler translates from textual format to binary format.

The question’s answer should clearly evidence knowledge of required salient facts relating to the two issues; human understandable textual format and machine understandable binary format; this was not done explicitly in some of the answers given. If the answers did not detail the differences plainly as stated in the above (and in the Example answer); e.g. answer gives evidence of knowledge of required salient facts, full marks were not awarded.

This theory of assemblers was covered in the lecture series in three different audio visual medias: 1) the actual live lecture; 2) the audio recording of the live lecture; and 3) the real time video of the lecture.

f) Explain the term ‘Garbage Collection’ and explain why it is important to implementations of Java. (4 marks)

Bookwork (4 marks):
The following points should be covered to some degree in the answer:

Programs which use dynamic memory allocation may make use of the memory for a period but then, after a time, use it no longer. In these circumstances, a Garbage Collector tries to identify dynamically allocated memory which is no longer used and returns it to the memory allocation system. In Java, objects are continually allocated and discarded, so GC is very important.

4 marks for an answer that depicts all the salient facts in a sensible way; and v. good briefly described;
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) [4]

Marker’s feedback 1.f.
Pedagogic assessment [criterion]:
The question assesses lecture 18 learning objective 8; define what is meant by [the term] Garbage collector; as the terms ‘live’ and ‘fragmentation’ are directly related to garbage collection.
Well done, the majority of you were able to explain the term ‘Garbage Collection’ and [then] explain why it is important to implementations of Java.
The question’s answer should clearly evidence knowledge of required salient facts relating to garbage collection and [could illustrate] the associated terms ‘Live’ and ‘Fragmentation.’ In the answer [some of] the following terminology (keywords and naming conventions [or sets of]) should be utilised in context; for example with respect to ‘Garbage collection’:

Page 3 of 15
memory, allocation, dynamically, no longer used, returns, allocated, discarded. Other related key words are: discovery of objects, stale objects, reference to an object, object is garbage. With respect to Fragmentation: holes, gaps, varying sizes, not marked live, objects removed. Main differentiation that must be clearly evidenced in your answer is that: Garbage Collection is a term directly related to objects still referenced (in use) are marked ‘live’ and are NOT garbage collected; whereas stale or non-live [or garbage] memory areas are returned to memory [where the stale object is removed from the memory]. The question’s answer should clearly evidence knowledge of required salient facts relating to garbage collection issues; this was not done explicitly in some of the answers given. If the answers did not detail the differences plainly as stated in the above (and in the Example answer); e.g. answer gives evidence of knowledge of required salient facts, full marks were not awarded. This theory of the three areas was covered in the lecture series in three different audio visual medias: 1) the actual live lecture; 2) the audio recording of the live lecture; and 3) the real time video of the lecture.
g) Explain how an expression of the form \( x = (a-b) \times (c+d) \) is evaluated using bytecode like instructions and write the code to show how the expression is evaluated. You may assume that \texttt{PUSH} and \texttt{POP} instructions are available but should define what they do.

(4 marks)

\textbf{Bookwork & Application (example) (4 marks):}

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

\textbf{Explanation:}

Bytecode instructions are zero address, which 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). \texttt{PUSH} and \texttt{POP} instructions move data between permanent memory and the stack.

The expression is evaluated thus:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operation [Function]</th>
<th>0- Or 1-address Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{PUSH a}</td>
<td>[Push on to stack] Move variable ‘a’ from memory to stack.</td>
<td>1-address Instruction</td>
</tr>
<tr>
<td>\texttt{PUSH b}</td>
<td>[Push on to stack] Move variable ‘b’ from memory to stack.</td>
<td>1-address Instruction</td>
</tr>
<tr>
<td>\texttt{SUB}</td>
<td>Subtract ‘a-b’; both on stack; resultant placed on stack [where ‘b’ was].</td>
<td>0-address Instruction</td>
</tr>
<tr>
<td>\texttt{PUSH c}</td>
<td>[Push on to stack] Move variable ‘c’ from memory to stack.</td>
<td>1-address Instruction</td>
</tr>
<tr>
<td>\texttt{PUSH d}</td>
<td>[Push on to stack] Move variable ‘d’ from memory to stack.</td>
<td>1-address Instruction</td>
</tr>
<tr>
<td>\texttt{ADD}</td>
<td>Add ‘a+b’; both on stack; resultant placed on stack [where ‘b’ was].</td>
<td>0-address Instruction</td>
</tr>
<tr>
<td>\texttt{MUL}</td>
<td>Multiply ‘(a-b) \times (c+d)’; both on stack; resultant placed on stack [where ‘(c+d)’ was].</td>
<td>0-address Instruction</td>
</tr>
<tr>
<td>\texttt{POP x}</td>
<td>[Pop off stack] Move resultant [answer] ‘(a-b) \times (c+d)’ from stack to memory.</td>
<td>1-address Instruction</td>
</tr>
</tbody>
</table>

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

\textbf{3 marks} for correct answer but not detailed [enough];

\textbf{2 mark} for a right-lines approach;

\textbf{1 mark} for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): Lecture 17: Java bytecode.

TOTAL marks (4 marks) [8?]
Marker’s feedback 1.g.

**Pedagogic assessment [criterion]:**

The question assesses lecture 17 learning objective 1, and 2 directly; 1. Gives examples of what [and why] Java uses Bytecode; [May be supported by a diagram!!!]; and 2. Explain the concepts behind Zero address [Instructions] [via writing byte code PUSH & POP instructions].

Well done, the bulk of you were able to explain how an expression of the form \( x = (a-b)*(c+d) \) is evaluated using bytecode.

First, what we would normally call keywords – should be re-cast as the appropriate code words or instructions – so - In the answer [some of] the following terminology (keywords and naming conventions [or sets of]) should be utilised in context; for example with respect to ‘evaluated using bytecode’ are: PUSH, SUB, ADD, MUL, & POP.

The real crux of the question involves accessing your ability to remember and write down in the correct sequence the appropriate keywords (instructions). Happily most of you achieved this and if your answer also aligned the correct variables \{a, b, c, d, & x\} to the correct instruction \{PUSH, SUB, ADD, MUL, & POP\} you were awarded full marks –well done.

However, if you did not state explicitly and correctly:
1. the correct instruction \{PUSH, SUB, ADD, MUL, & POP\} names;
2. or write the correct variables \{a, b, c, d, & x\} next to the correct instruction;
3. or the sequential order of the instructions was wrong – re. template answer – full marks could not be awarded; a few of you did indeed make mistakes with respect to 1. 2. & 3 and hence marks were not awarded.

This theory of the three areas was covered in the lecture series in three different audio visual medias: 1) the actual live lecture; 2) the audio recording of the live lecture; and 3) the real time video of the lecture.
Section B

3.

a) What is the ‘heap’ in a Java Virtual Machine? Your explanation should include an explanation of when it is used and what is stored there.

(4 marks)

Bookwork (4 marks):
The following points should be covered to some degree in the answer:

Explanation: The heap is an area of permanent memory which is allocated dynamically when needed.

When it is: Its usual use in the implementation of Java is to store information associated with an instance of an object. The memory is allocated when a ‘new’ is executed.

What is stored there: The major content of the memory allocated are the instance variables of the object. However, there is also extra information associated with the class, size and structure of the object.

4 marks for an answer that depicts all the salient facts in a sensible way; all keywords correctly delineated [in context],
3 marks for correct answer but not detailed [enough]; three steps,
2 mark for a right-lines approach; two steps,
1 mark for some basic understanding (or attempt); one step.
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): Lectures 18: Java Memory Usage.

TOTAL marks (4 marks) [4]

Marker’s feedback 3.a.
Pedagogic assessment [criterion]:
The question assesses lecture 18 learning objective 5; 5. Explain the function of the Heap. Well done, most could explain what a ‘heap’ is in a Java Virtual Machine. The question’s answer should clearly evidence knowledge of required salient facts relating to the term ‘heap.’
In the answer [some of] the following terminology (keywords and naming conventions [or sets of]) should be utilised in context; for example with respect to ‘heap’: area, permanent, memory, allocated, dynamically, to store, information, associated, instance, object, ‘new’, executed, memory, instance, variables, object, class, size, structure.
Main differentiation that must be clearly evidenced in your answer is that: a heap is a term directly related to an area of memory; and evidence correct usage of above keywords in the correct context.
The question’s answer should clearly evidence knowledge of required salient facts relating to heap issues; this was not done explicitly in some of the answers given. If the answers did not detail the explicit theory plainly as stated in the above (and in the Example answer); e.g. answer gives evidence of knowledge of required salient facts, full marks were not awarded. This theory of the three areas was covered in the lecture series in three different audio visual medias: 1) the actual live lecture; 2) the audio recording of the live lecture; and 3) the real time video of the lecture.
b) Describe the concept of a stack and discuss why it is useful in the implementation of subroutine calls. (4 marks)

**Bookwork (4 marks):**

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

**Description:** A stack is like [analogous to] a pile of coins implemented as a storage structure in memory. Or, a list where insertion and deletion takes place at one of the ends of the list is called a stack. We put things on the top and take them off the top.

Why it is useful [for subroutine calls]: If we perform nested subroutine calls in a program, it is necessary to remember register values which are re-instated after the call. The order of these is **FIFO** and therefore a stack is ideally suited to this. It is also possible to use the stack to pass parameters and provide temporary storage.

**4 marks** for an answer that depicts all the salient facts in a sensible way; all keywords correctly delineated [in context],

**3 marks** for correct answer but not detailed [enough]; three steps,

**2 mark** for a right-lines approach; two steps,

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

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

TOTAL marks (4 marks) [8]

**Marker’s feedback 3.b.**

**Pedagogic assessment [criterion]:**
The question assesses lecture 18 learning objective 3; 3. Identify the 3-basic java areas of storage;

Well done, in the main most were able to describe the concept of a stack and discuss why it is useful in the implementation of subroutine calls.

The question’s answer should clearly evidence knowledge of required salient facts relating to the term ‘stack.’

In the answer [some of] the following terminology (keywords and naming conventions [or sets of]) should be utilised in context; for example with respect to ‘stack’: pile of coins, implemented, storage, structure, memory, put things on the top, take them off the top, remember register values, re-instated after the call, FIFO, pass parameters, temporary storage, area of memory.

The question’s answer should clearly evidence knowledge of required salient facts relating to stack issues; this was not done explicitly in some of the answers given. If the answers did not detail the explicit theory plainly as stated in the above (and in the Example answer); e.g. answer gives evidence of knowledge of required salient facts, full marks were not awarded.
This theory of the three areas was covered in the lecture series in three different audio visual medias: 1) the actual live lecture; 2) the audio recording of the live lecture; and 3) the real time video of the lecture.
c) A ‘stack’ program is depicted in figure 3.c.1.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
<th>Label</th>
<th>Mnemonic</th>
<th>R0</th>
<th>R13 [SP]</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>000</td>
<td>StackTop2</td>
<td>DEFW 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>004</td>
<td>StackStart2</td>
<td>DEFW 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>008</td>
<td>StackBottom2</td>
<td>DEFW 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00C</td>
<td>StackProg2</td>
<td>MOV R0, #1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>010</td>
<td></td>
<td>ADR SP, StackStart2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>014</td>
<td></td>
<td>STR R0, [SP]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>018</td>
<td></td>
<td>MOV R0, #2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01C</td>
<td></td>
<td>STR R0, [SP, #4]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>020</td>
<td></td>
<td>SVC 2 ; (or SWI 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Describe in detail exactly what happens when the above ARM program is obeyed; using the table below. In the table clearly describe the movement of information (both numbers and instructions) between memory, stack and the CPU in the comments column, and how the values in the registers R0, and R13 [SP] change, at each step.

<table>
<thead>
<tr>
<th>Address/ label</th>
<th>Mnemonic</th>
<th>R0</th>
<th>R13 [SP]</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>00C</td>
<td>MOV R0, #1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>ADR SP, StackStart2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>014</td>
<td>STR R0, [SP]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>018</td>
<td>MOV R0, #2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01C</td>
<td>STR R0, [SP, #4]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>SVC 2 ; (or SWI 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assume that the program starts at memory location 0x00C; for this exam question.

In your answer please draw up a table, similar to the one above, which has the five columns for: addresses, mnemonics, registers and the comments in your answer book when you answer the question.

(8 marks)
Marker’s feedback 3.c.

Pedagogic assessment [criterion]: Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Laboratory exercise 4, Lecture 9: Methods and Stacks & Lectures 18: Java Memory Usage – in particular learning outcome: 3. Identify the 3-basic java areas of storage.

Well done, a large proposition was able to describe in detail exactly what happens when the above ARM program is obeyed; using the table - figure 3.c.2.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Registers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add/label</td>
<td>Mnemonic</td>
<td>R0</td>
</tr>
<tr>
<td>00C</td>
<td>MOV R0,#1</td>
<td>0x0000,0001</td>
</tr>
<tr>
<td>010</td>
<td>ADR SP,StackStart2</td>
<td>0x0000,0001</td>
</tr>
<tr>
<td>014</td>
<td>STR R0,[SP]</td>
<td>0x0000,0001</td>
</tr>
<tr>
<td>018</td>
<td>MOV R0,#2</td>
<td>0x0000,0002</td>
</tr>
<tr>
<td>01C</td>
<td>STR R0,[SP, #4]</td>
<td>0x0000,0002</td>
</tr>
<tr>
<td>020</td>
<td>SVC 2 [or SWI 2]</td>
<td>0x0000,0002</td>
</tr>
</tbody>
</table>

Caveat – note: JG re-evolution of the model answer has all the above numbers in the next rows down. This is because in Reality the CPU [ARM] goes through the fetch-decode-execute cycle so the data is only valid after the instruction has run. So really the answers [numbers] should all be one row down. But if you state “ASSUMPTION: third row R0 and fourth row R13 – show data after instruction has executed.” As the result you would see with Komodo would indeed only change the R0 & R13 registers after the instructions have executed; so in actuality would be one row down; e.g. 0x0000,0001 would appear in row 010 not 00c…

8 marks for all three [extra] columns [totally] correct,
4 marks for two of the columns correct,
2 marks for 1 one column, or some errors.

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Laboratory exercise 4, Lecture 9: Methods and Stacks & Lectures 18: Java Memory Usage.

TOTAL marks (8 marks) [16]
This type of exercise – e.g. in Example Sheet 3, & Exercise Sheet 4 – has been one of the methods you would have used to comprehend how a machine code [assemble language] program works on paper.

The crux of it requires some understanding of keywords and machine [assembly language] instructions, e.g. [what explicitly a]: MOV, ADR, STR, & SVC; instructions are and what they explicitly do. It also require some thinking back to when you undertook some of the laboratory exercises – where you will have monitored what happened in the registers [on the left hand side of the Komodo simulator]. You should have – by the exam time – had quite a bit of practice – with respect to writing verbose comments to assembly code instructions – as you will have experienced it in: lectures, laboratory exercise classes, and exercise class in tutorials – plus if you have scanned the past exam papers you will have seen this is a skill that is [and has been] assessed in exams.

Some, however, were either: too fuzzy, or not explicit enough, or did not fill in column for R0 and R13 correctly – and may have not put the numbers in each row – as required – hence marks could not be awarded.

This theory of the three areas was covered in the lecture series in three different audio visual medias: 1) the actual live lecture; 2) the audio recording of the live lecture; and 3) the real time video of the lecture.
d) Given the stack pointer ARM program above, in question 3.c. Draw up a copy of the table in figure 3.d. in your answer book, and fill in the appropriate data. Assume that the program has been executed; for this exam question. In the table clearly state: the memory addresses, stack data and the stack pointer position after the program has executed. (4 marks)

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Memory data</th>
<th>Position of stack pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x----,----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x----,----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x----,----</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question figure 3.d. Stack Pointer Diagram, composed of three columns: Memory Address, Memory data, and Position of stack pointer columns.

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

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Memory data</th>
<th>Position of stack pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000,0000*</td>
<td></td>
<td>← SP*</td>
</tr>
<tr>
<td>0x0000,0004*</td>
<td>01*</td>
<td></td>
</tr>
<tr>
<td>0x0000,0008*</td>
<td>02*</td>
<td></td>
</tr>
</tbody>
</table>

4 marks for all six items * correct,
2 marks for four correct, etc
1 marks for 2, or some errors.
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Laboratory exercise 4, Lecture 9: Methods and Stacks & Lectures 18: Java Memory Usage.

TOTAL marks (4 marks) [20]

Marker’s feedback 3.d.
Pedagogic assessment [criterion]: Laboratory exercise 4, Lecture 9: Methods and Stacks & Lectures 18: Java Memory Usage; and explicitly learning outcome: 3. Identify the 3-basic java areas of storage.
Well done, most students were able to draw up a Stack Pointer Diagram, composed of three columns: Memory Address, Memory data, and Position of stack pointer; as depicted in figure 3.d. Quite a few forgot to put the SP or explicitly “← SP*” in the third column marked “Position of stack pointer”; so were marked down – as it was explicitly asked for in the question and a column was provided in the template answer.

The stack theory was covered in lectures Lecture 9: Methods and Stacks – in the first half of the course – so there was supporting theory to assist you here.

It was also covered in the second half of the course in Lectures 18: Java Memory Usage; and explicitly [covered in] learning outcome: 3. Identify the 3-basic java areas of storage.

The marks showed most were able to work out the correct:
1. memory stack addresses;
2. memory stack data; and finally
3. the actual position of the stack pointer.

If you did all the above {1, 2, & 3} well done.

However a few did not get one of the three {1, 2, & 3} or two of the three {1, 2, & 3} correct and then marks could not be awarded.

This theory of the three areas was covered in the lecture series in three different audio visual medias: 1) the actual live lecture; 2) the audio recording of the live lecture; and 3) the real time video of the lecture.

END OF EXAMINATION