Javier Navaridas-Palma:

The general impression about the exam is good, it seems that students got quite good performance in the exam. Part C seemed to be more difficult for students than the rest of the exam. In this part, most student's didn't show a working understanding of how to use data structures (arrays, stack). A small part wasn't able to construct basic control structures (if, while) properly. There were also problems differentiating basic concepts such as instruction, pseudoinstruction or directive.

Part A was the easiest and, in general, students got good marks. Some common error was to confuse 1-address instruction style with stack-processing and defining a stack as a FIFO structure. Some students also thought that the stack was implemented in registers, rather than in main memory. Questions C and D, were generally good. I accepted as good all answers where the process was adequate, even if they had calculation errors.

Richard Neville:

Please see attached pdf.
Section A

THREE short questions from RN

General Feedback Comments
The following general comments are suggested to make your reflection and feedback more readable/succinct and viable for a more general succinct meditation on what you could do to enhance your learning and may be adapt your revision methodology.

First, it is important to reflect on the last lecture RN presented to the cohort; at the end of that lecture a comprehensive set of steps and guidance was presented for revision that students have advised me that they utilise. This was derived from methods students have utilised to revise over the years. The sentiments and guidance in this revision was sometimes given in their own words [the students]; and their own reflection on what worked best for different situations. But, may be it is worth noting that sometimes there can be discrepancies between the student’s view of what mark they should have attained and what they actually were awarded. Reflecting on this issue may be it is worth noting and quoting the specific feedback from a student with respect to using the [suggested] self-test (or self-assessment) methodology, they said:

“With regards to the self-assessment questions definitely allowed me to retain and recall large amounts of domain knowledge.
It was especially useful in shorter questions and proved more beneficial than simply re-reading notes in a repetitive manner.”

This is pertinent as without utilising a method like the self-assessment as well as undertaking a number of past exam papers one cannot self-access one’s ability to pass the exam or access what mark one may obtain; and it is even more important when one goes into industry as without a good understanding of your own abilities how can you decide which courses to take, either those presented by the company that employs you or by external courses, and hence how can you evolve your personal skill set. You could also assess where you feel you are in the The Four stages of Learning (4SoL): or Do you know what you know?, information on this [4SoL] will be placed on Blackboard 9 soon.

Good companies will encourage you to undertake CPD. Continuing professional development (CPD) or Continuing professional education (CPE) is the means by which people maintain their knowledge and skills related to their professional lives.

One could say that evolving your revision and exam skill at University is, in fact, a form of CPD.

A final point, before getting into the detailed feedback for each question, is to reflect or ask yourself questions such as:
Did I undertaking past exam papers? [enough]
Did I undertaking a past exam paper – timed? [to get used to the time constraint of a real exam]
Did I develop a self-test (or self-assessment)? and finally
Did you utilise the methodologies presented in RN’s exam revision lecture?

One of the recurring points was the advice on diagrams does not seem to be have been noted by many students; hence we repeat it hear: Remember, good – honours grade answer – in the exam – for a question – to maximise marks – should – or you should think of adding A DIAGRAM or a set of diagrams; hence a basic layout of a question answer may be:
1) Textual answer;
2) Diagram supporting answer [or code snippet]; &
3) Full explanation of diagram…

This sort of answer will [may] maximise your marks…
1. **Computer Architecture**

   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)

1.e.

**Bookwork (2 marks):**

i) What mechanism is used to achieve this *mapping*? (1 mark)

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

The mechanism is an interrupt vector table.

ii) Briefly explain this mechanism? (1 mark)

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

- The vector table is [composed of] an array of pointers to handlers; and the processor uses the interrupt number as its index into this array.
- Alternatively: The Interrupt vector table consist of [or is composed of] designated addresses in external memory that hold information necessary to handle an interrupt.

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 14: Input/Output (2).

TOTAL marks (2 marks) [2]
Marker’s feedback 1.e.

**Pedagogic assessment [criterion]:**
The question assesses lecture Lecture 14: Input/Output (2) objective 1: Distinguish between ARM’s interrupts [interrupt vector table].

Well done, most of you were able to state what mapping was performed and you stated it was performed by the interrupt vector table mechanism.

The question’s answer should clearly evidence knowledge of required salient facts relating to the function of an interrupt vector table.

In the answer [some of] the following terminology (keywords and naming conventions) should be utilised in context; for example: vector table, pointers, handlers, interrupt number, index, array.

Main differentiation that must be clearly evidenced in your answer is that: vector table mechanism performed a translation, or lookup – given the interrupt number it uses this as an index into the interrupt vector table – that then looks up the appropriate interrupt handler routine – which services the interrupt.

The question’s answer should clearly evidence knowledge of required salient facts relating to the issues aligned to interrupt vector table; this was not done explicitly in some of the answers given. If the answers did not detail the mechanism 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 interrupt vector table 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) 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; using the table in figure 1.f. 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>Operand</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR</td>
<td>R1, Acknowledge</td>
<td>:Load R1 with the Acknowledge address.</td>
</tr>
<tr>
<td>LDR</td>
<td>R0, #1</td>
<td>:Load R0 with #1 to acknowledge read.</td>
</tr>
<tr>
<td>STR</td>
<td>R0, [R1]</td>
<td>:Store the acknowledge where R1 points.</td>
</tr>
</tbody>
</table>

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

1.f.
Application (Critique) (4 marks):

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 what happens when the ARM program is obeyed; using the table in figure 1.f. 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)

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

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 13: Input/Output (1).

TOTAL marks (4 marks) [6]
Marker’s feedback 1.f.

Pedagogic assessment [criterion]:
The question assesses Lecture 13: Input/Output (1) objective 4; Illustrate why interrupts are used instead of polling – as this question is related to this issue.

Well done, the majority of you were able to explain the issue of a ‘code snippet from a very simple interrupt vector handler’ and [then] explain each line in the program. The question’s answer should clearly evidence knowledge of required salient facts when commenting the code snippet from a very simple interrupt vector handler.

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 commenting the ‘code snippet from a very simple interrupt vector handler’: load, acknowledge, read, address, store, & points.

Main differentiation that must be clearly evidenced in your answer is that: your comments are directly related to each instruction – and the operation it performs – and the registers, & addresses involved.

The question’s answer should clearly evidence knowledge of required salient facts relating to each instruction and the issue it is dealing with {loading, storing, pointing etc.}; this was not done explicitly in some of the answers given. If the answers did not detail the differences plainly [between each instruction] 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 base plus index addressing as implemented in the ARM processor.

Bookwork & explanation (4 marks):

Base plus index addressing is a form of indirect addressing where the address in memory is formed by adding or subtracting the contents of two registers. One is normally assumed to point to the base of a data structure while the other holds an offset [index] within that structure.

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 19: Arrays (1): Array Implementation.

TOTAL marks (4 marks) [10]
Marker’s feedback 1.g.

Pedagogic assessment [criterion]:
The question assesses Lecture 19: Arrays (1): Array Implementation learning objectives:
3. Differentiate between: reference pointer, array base, array index;
4. Specify methodology of indexing into an array;
4.a. Also state how base & index are used in conjunction; and
4.b. Also posit how index mathematics is undertaken.

Well done, the bulk of you were able to explain how the base and indexing mechanism works – and the theory behind it.
First, what we would normally call keywords: Base, plus, index, addressing, indirect, memory, adding, or subtracting, two registers, and data structure.

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 theory you were awarded full marks – well done.

However, if you did not state explicitly and correctly:
1. the majority of the appropriate keywords in context – and plain English;
2. or you wrote an incomplete description – not including the majority of keywords or the majority of the theory;
3. or the answer was fuzzy and not in Plain concise English – 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 area 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) Name the four steps and briefly describe each step of an assembler. (4 marks)

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

Step 1) **Lexical** (word) analysis; the process of converting a sequence of characters into a sequence of tokens;
Step 2) **Syntactic** (structure) analysis; checking instructions are legal;
Step 3) **Semantic** (meaning) analysis, check user-defined names - declared exactly once; and
Step 4) **Code generation**; translate to binary machine code.

4 marks for majority of above; e.g. for an answer that mentions the salient facts in a sensible way (2 marks for a detailed and concise description.),
3 marks for correct answer but not detailed [enough],
2 mark for brief explanation & 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 16: Assemblers and Compilers.

TOTAL marks (4 marks) [4]

**Marker’s feedback 3.a.**

**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.
b) Given the ARM code snippet:

```
start LDR R0, a
```

A set of keywords aligned to the assembly steps are: values; discard spaces; instructions; character; list; space; names; word; sequence; tokens; and legal.

i) Align the appropriate keywords above to the correct assembly step (you named in 3.a) in a table drawn up in your answer book similar to that depicted in figure 3.b.; whilst also naming the assembly steps – in the first column.

<table>
<thead>
<tr>
<th>Assembly step</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1:</td>
<td></td>
</tr>
<tr>
<td>Step 2:</td>
<td></td>
</tr>
<tr>
<td>Step 3:</td>
<td></td>
</tr>
<tr>
<td>Step 4:</td>
<td></td>
</tr>
</tbody>
</table>

Question figure 3.b. A table; showing four steps in the assembly process.

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

iii) State the three [hierarchical] abstraction levels of programming languages; also state which is human and which machine understandable. (3 mark)
3.b.i. Application (example) (Critique) (4 marks):

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

i) Align the appropriate keywords above to the correct assembly step (you named in 3.a);

<table>
<thead>
<tr>
<th>Assembly step</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: Lexical (word) analysis</td>
<td>character; word; space; discard spaces;</td>
</tr>
<tr>
<td></td>
<td>sequence; tokens</td>
</tr>
<tr>
<td>Step 2: Syntactic (structure)</td>
<td>instructions; legal</td>
</tr>
<tr>
<td>step analysis</td>
<td></td>
</tr>
<tr>
<td>Step 3: Semantic (meaning)</td>
<td>list; names; values;</td>
</tr>
<tr>
<td>analysis</td>
<td></td>
</tr>
<tr>
<td>Step 4: Code generation</td>
<td>-</td>
</tr>
</tbody>
</table>

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).: Lectures 16: Assemblers and Compilers.

TOTAL marks (4 marks) [8]

Marker’s feedback 3.b.i.

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 keywords were associated with the appropriate step of an assembler.

The question’s answer should clearly evidence knowledge of required salient facts relating to the function of an assembler and comprehension of the appropriate keywords.

In the answer [some of] the following terminology (keywords and naming conventions) should be utilised in context; for example: Lexical, Syntactic, Semantic, Code, analysis, generation.

Main differentiation that must be clearly evidenced in your answer is that: the assembler steps are all named and the appropriate order and row; and that the keywords alignment was correct – alignment to correct step.

The question’s answer should clearly evidence knowledge of required salient facts relating to the two issues: naming steps and aligning stated (given) keywords to appropriate steps; 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.
3.b.ii.
Application (example) (3 marks):

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

ii) A compiler has five steps; name and briefly describe the compiler's last step; also give examples of how it achieves the fifth step's goal.

(3 mark)

The compiler's fifth and final step is: Code optimisation:
This is performed in order to optimise the translations for: speed, memory [usage] – examples of how it achieves its goal are:

i. keep variables in registers;
ii. move code out of loop bodies; and
iii. use "clever" instructions (conditional instructions).

3 marks for an answer that depicts all the salient facts in a sensible way; correctly delineated [in context],
2 mark for a right-lines approach; but not as detailed,
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): Lectures 16: Assemblers and Compilers.
TOTAL marks (3 marks) [11]

Marker’s feedback 3.b.ii.

Pedagogic assessment [criterion]:
The question assesses lecture 16 learning objective 2, 3 and 4 indirectly; 2. Differentiate between an Assembler and a Compiler; 3. Summarise the five Compiler steps; and Contrasts the Compiler steps [in detail].
Well done, most of you were able to state what function the final step of the compiler performs - it performs the optimization 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 a compiler in the optimization 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: code, optimisation, speed, memory, keep, variables, registers, move, code, out of loop, clever, instructions, conditional instructions.
Main differentiation that must be clearly evidenced in your answer is that: the compiler optimised the code format to binary format with fewer instructions.
The question’s answer should clearly evidence knowledge of required salient facts relating to the two issues; comprehension of fifth step – that of optimisation - and examples of how to explicitly achieve this; 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.
3.b.iii.
Application (example) (4 marks):

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

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

Three “abstraction levels” of Programming Languages are:

1. Binary Machine Code; is Machine understandable;
2. Assembly Language; is Low-Level Language;
3. High-Level Language; is Human understandable.

3 marks for an answer that depicts all the salient facts in a sensible way; and correctly delineated [in context],
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).: Lectures 16: Assemblers and Compilers.
TOTAL marks (3 marks) [14]

Marker’s feedback 3.b.iii
Pedagogic assessment [criterion]:
The question assesses lecture 16 learning objective 1, Distinguishes between the abstraction levels of programming languages.
Well done, most of you were able to state what the abstraction levels were.
The question’s answer should clearly evidence knowledge of required salient facts relating to the different abstraction levels.
In the answer [some of] the following terminology (keywords and naming conventions) should be utilised in context; for example: Binary, Machine, Code, Machine, understandable, Assembly, Low-Level, Language, High-Level, Human.
Main differentiation that must be clearly evidenced in your answer is that: clearly and explicitly differentiate and name the three levels.
The question’s answer should clearly evidence knowledge of required salient facts relating to the two issues; the explicit and correct names of each level and a brief description of each; 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.
c) What is bytecode in the context of Java and how does it get executed by a real processor? (3 marks)

3.c.
Bookwork (Critique) (3 marks):

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

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

1) The Java compiler does not produce real machine instructions.
2) It produces instead, bytecode, which is a stack based (zero address) instruction set.
3) This has the advantages of portability and compactness.
4) The simplest way to execute bytecode is with a software interpreter.
5) However, performance considerations have led to the development of dynamic compilation virtual machines.
6) Here bytecode is initially interpreted but, if monitoring shows that the code is being heavily executed, the bytecode is translated to native machine code.

3 marks for all six points – covered to some degree,
2 marks for three to four points – covered to some degree,
1 marks for 1 a few, and some errors.
Reference Learning Resources, Background Reading, and Lecture itself for detailed information;
Lecture 17: Java bytecode.
TOTAL marks (3 marks) [17]

Marker’s feedback 3.c.
Pedagogic assessment [criterion]:
The question assesses lecture 17 learning objective 1, Gives examples of what [and why] Java uses Bytecode; [May be supported by a diagram!!!].
Well done, most of you were able to state what function of bytecode.
The question’s answer should clearly evidence knowledge of required salient facts relating to byte code in the context of Java.
In the answer [some of] the following terminology (keywords and naming conventions) should be utilised in context; for example: Java compiler, produce, real, machine, instructions, stack, based, zero address, instruction set, portability, compactness, execute, software interpreter, performance, dynamic compilation, & virtual machines.
Here bytecode is initially interpreted, monitoring, code, heavily executed, translated, native machine code.
Main differentiation that must be clearly evidenced in your answer is that: the byte code description covers all the appropriate issues – as presented in the lecture series, text books, and video.
The question’s answer should clearly evidence knowledge of required salient facts relating to the two issues; explicitly what is bytecode and in a broader sense bytecode in the context of Java; plus when knowledge aligned to the fact that is first interpreted – then if heavily executed sections [of code] are repeated they are converted to machine code. 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.
d) Name the three Operating Systems kernel managers and briefly describe each. (3 marks)

3.d. 
Bookwork (Critique) (3 marks):

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

The three-managers and descriptions are:
1. **Process** Manager (CPU): creates the illusion that several things can be going on at once.
2. **Memory** Manager (main memory): each program has all the memory it needs, but can’t access anything else.
3. **Peripheral** Managers: separate manager (device driver) for each (kind of) peripheral.

**3 marks** for an answer that depicts all the salient facts in a sensible way; all three managers correctly delineated and briefly described;
**2 marks** for correct answer but not detailed [enough];
**1 mark** for a right-lines approach;
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lecture 15: Interrupts.
TOTAL marks (3 marks) [20]

**Marker’s feedback I.e.**
**Pedagogic assessment [criterion]:**
The question assesses lecture 15 learning objective [LO’s]; 2) List five of the main functions a Kernel performs for an (OS) ; & 3) Recognise the provision the kernel provides;

The question’s answer should clearly evidence knowledge of required salient facts.

In the answer [some of] the following terminology (keywords and naming conventions) should be utilised in context; for example: **Process**, several, processes [things], [on at] once [concurrently], **Memory** (main memory, program, memory, access [anything else], **Peripheral**, Managers, device driver, peripheral.

Well done, most students were able to mention and then describe [in some technical matter] the 3 key items: Process, Memory, & Peripheral Managers.
The marking differentiation [the difference between high and low marks] was due to; the following:

1/ All three were correctly named and explicitly described – using the correct technical terminology.

2/ If the terminology was not technically correct marks were dropped [not awarded] – this happened in the case of some students as their answers depth of coverage was not technically correct – and did not utilise the correct technical naming conventions.

3/ Other students with lower marks [whom were awarded lower marks]; either because:

   3.1/ They named only 2 or 1 out of the 3 managers; or

   3.2/ Named only one correctly and either gave a too brief description – or the description was not technically correct.

The other key issues where students did not gain marks were in the technical descriptions of some of the managers. Some student’s answers were very brief and did not even cover the issues template in the marking scheme answers – hence marks could not be awarded.

Fuzzy in the context of grammar, technical terminology or the fact that they were not concise answers also meant marks were not awarded.

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.

END OF EXAMINATION