THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2000

EXAMINERS' REPORT

System Software

General Comments

There were only a small number of candidates for this paper and the pass rate attained was not unreasonable. The examiners had expected that candidates for this paper would have preferred the 'doing' questions (Q1 and Q3) rather than the talking questions. They were disappointed that this was not so.

Question 1 – Answer Pointers

Most block-structured high-level programming languages permit a procedure to be declared within another procedure.

Consider such a case, in which a procedure named INNER is declared inside another named OUTER:

both are recursive;
both INNER and OUTER have parameters and variables;
a variable V, declared in OUTER, is also referred to by the instructions in INNER.

When OUTER has recursed, there will be an instance of V for each instance of OUTER. Assume that at such a time a reference to V from an instance of INNER will be to the V of the most recent instance of OUTER.

Describe how the run-time stack can be organised to support this language capability. [15 marks]

Include how parameters and other variables of both procedures would be incorporated. [10 marks]

Material can be found in Gries chapters 8 and 14.

The question requires moderate understanding of run-time stacks, and (unless it is known as bookwork - not in Gries) a touch of creativity for the access to OUTER vartiables.

Framed stack - concept 5 marks, parameters & return links 5 marks, local variables 5 marks, frame chain for recursive procedures, 10 marks.

There was only one answer to this question

Question 2 – Answer Pointers

How is virtual memory implemented for *either* a paged system *or* a segmented system?

Describe the necessary hardware aspects and the data structures used, and the ways in which those structures are employed for address translation and for memory management.

Illustrate your answer with examples.

A good answer would have included the concept of fixed-size pages (virtual, mostly in backing store) matched with page frames in memory, where the former outnumbered the latter. Addresses needed to be translated, using page tables with appropriate contents (with hardware assistance). The details of these were required, to demonstrate a proper understanding. The implications of page faults, protection, and placement algorithms were also required for a full answer. The marking scheme was as follows:

Hardware requirements: associative memory, addressing and protection registers to match. Use thereof. *[8 marks]*

Data structures: page / segment tables, and their use. *[7 marks]*

Illustrations: translation of virtual to real addresses, page / segment faults, protection, and the treatment of each case. *[10 marks]*

This was a popular question, but not all of the candidates were as familiar with the processes as was hoped for. Candidates chose to describe segmentation rather than paging. They showed a general understanding of the principles but the actual mechanisms used were not always well described.

Question 3 – Answer Pointers

a) Using a high-level programming language, write a sequence of statements using a single loop which examines the components of a one-dimensional array of integers with an arbitrary range of indexes N to M, summing the values at indexes divisible by four into a variable named SUMFOUR and summing the values at other indexes into a variable named SUMOTHER.
[5 marks]

b) By writing assembler code for an actual or fictitious CPU, show how the compiler could optimise the test for "divisible by four". [5 marks]

b) Suppose that in the code you wrote for part a, the compiler can determine that N=1 and that M is divisible by four. By re-writing the high-level code, and (if needed) by writing assembler code equivalent to key parts of it, explain how the compiler can optimise the loop as a whole to maximise speed.
[15 marks]

The question addresses syllabus sections 5 and 6. Material can be found in Gries chapter 18.

(a) something like

```
SUMFOR := 0;
SUMOTHER := 0;
for I in N..M loop
   if I rem 4 = 0
   then SUMFOR := SUMFOR + A(I);
   else SUMOTHER = SUMOTHER + A(I);
   end if;
end loop;
```

[5 marks]

(b) - a strength reduction from `I rem 4 = 0` *to* `I bitwise-and 3 = 0`.
[5 marks]

(c)

```
SUMFOR := 0;
SUMOTHER := 0;
M4 := (M rightshift 2);
I := 1;
for J in 1..M4 loop
   SUMOTHER = SUMOTHER + A(I++) + A(I++) + A(I++);
   SUMFOR := SUMFOR + A(I++);
end loop;
```

Marks for

- *the 4-way unrolling*
- *the strength-reduction rightshift for division by 4 (in a way, part b is a clue for this)*
- *the precession of I.*

[15 marks]

Other patterns of answer - especially if in assembler, or part-assembler - would have been acceptable.

Disappointingly, there was only one attempt at this question and that was derisory.

Question 4 – Answer Pointers

Although the hardware interfaces of different peripheral devices differ, it is desirable for uniform software interfaces to be provided for them. Describe, with examples, typical hardware and software interfaces.

An understanding of device drivers is required for this question. This part expects descriptions of the information typically coming from a number of hardware devices; a spread of such devices will be needed to give sufficient diversity and therefore justify the need, so that at least one input and one output device would be needed. A good treatment of one of each could earn full marks for this part, although a not-quite-so-good attempt using more devices would also be accepted. *[10 marks]*

What mechanisms are provided to connect the two types of interface for a given hardware device?

An outline of the device driver concept with some suitable arguments for their existence was expected; no great detail was necessary for marks in this part. *[5 marks]*

Describe the activities typically carried out by such a mechanism.

This part deals with what happens inside a device driver: ports, bits, specialised instructions, use of interrupts, timing, etc.; extreme accuracy will not be needed (although it would be welcome), but a solid understanding is required.[10 marks]

An number of appropriate hardware interfaces were discussed by the candidates in this popular question, but not as many relevant software interfaces. The connection between the two was understood by a minority of the answers only; and therefore the activities which such mechanisms (typically, device drivers) are expected to carry out were less well addressed than was hoped for.

Question 5 – Answer Pointers

Explain why an interpreted program will run much slower than a corresponding compiled program. [25 marks]

This question addresses syllabus section 5. Material may be found in Gries chapters 16 to 18.

Straightforward bookwork. The higher marks were given to answers that were clearer and had enough detail to demonstrate understanding.

More popular than the other compiler/language questions (Q1 & Q3) presumably because it is a "talk" question, whereas Qs 1 & 3 are "do". A couple of answers showed lack of knowledge - why choose this question? Most showed extensive knowledge, but varied in how well it was applied to the required issue.

Question 6 – Answer Pointers

List and describe in outline the components which make up an operating system of your choice, which should be named.

Using suitable examples, show how these components interact with each other to satisfy the users' requirements.

The question is standard bookwork: the first part expects a choice from a wide range of components. These include memory allocators and managers, low level schedulers, device handlers, windows managers, command line interpreters, and so on. Full marks were given for six components.

[10 marks]

The more important part of the question should describe how these items interact. Sensible diagrams are hoped for. Ten marks were given for a clear explanation and five for suitable examples.

[15 marks]

This question was attempted by all the candidates, mostly with reasonable success. Many possibilities were available and candidates collectively achieved a wide spread. There was confusion about operating system components in some cases, but many were well-categorized. The formal interactions between the components were not always included. Examples given were of varying quality.