BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 4 Certificate in IT

COMPUTER & NETWORK TECHNOLOGY

Examiner Report

March 2018

General comments on candidates' performance and the process

The most popular question was A1 (73.74%), followed by A2 (50.11%), A4 (37.86%) and A3 (30.20%) – thus, the most popular combination was A1+A2.

Answer pointers

A1 Part (a) 8 marks

A logic circuit has four binary inputs D, C, B, A that represent the sixteen values 0 (0,0,0,0) to 15 (1,1,1,1). The output F is true if the input on D, C, B, A is divisible by 3, 8, 13, or 14. Note that 0 is not divisible by any number. Draw a truth table for this system with inputs D, C, B, A and output F.

Part (a) Truth table

D	C	B	A	Number	Div 3	Div 8	Div 13	Div 14	F
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	0	1	0	2	0	0	0	0	0
0	0	1	1	3	1	0	0	0	1
0	1	0	0	4	0	0	0	0	0
0	1	0	1	5	0	0	0	0	0
0	1	1	0	6	1	0	0	0	1
0	1	1	1	7	0	0	0	0	0
1	0	0	0	8	0	1	0	0	1
1	0	0	1	9	1	0	0	0	1
1	0	1	0	10	0	0	0	0	0
1	0	1	1	11	0	0	0	0	0
1	1	0	0	12	1	0	0	0	1
1	1	0	1	13	0	0	1	0	1
1	1	1	0	14	0	0	0	1	1
1	1	1	1	15	1	0	0	0	1

To construct the truth table, 1 has been entered in the column of each of the four specified conditions. A 1 is put in the output column F if and of the four conditions is true.

A1 Part (b) 8 marks

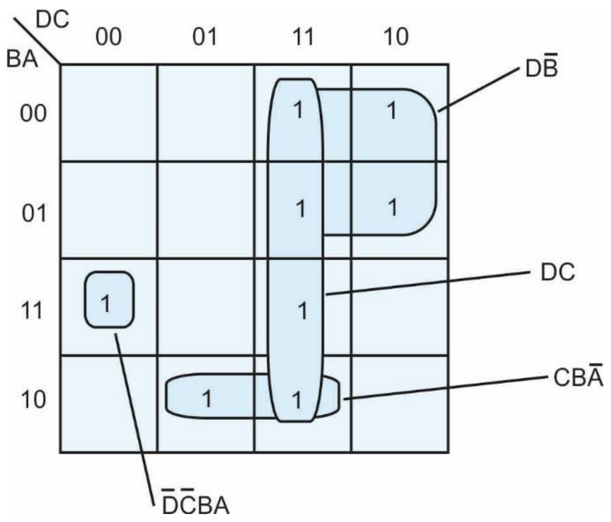
From the truth table, write down an expression for the output F (unsimplified).
F is given by the sum (logical OR) of all condition that set F to 1. That is
F = D!C!BA + D!CBA! + DC!B!A! + DC!B!A + DCB!A! + DCB!A + DCBA! + DCBA

A1 Part (c) 8 marks

Simplify this expression
F = D!C!BA + D!CBA! + DC!B!A! + DC!B!A + DC(B!A! + B!A + BA! + BA)
F = D!C!BA + D!CBA! + DC!B!(A + A!) + DC (using B!A! + B!A + BA! + BA = 1)
F = D!C!BA + D!CBA! + DC!B! + DC
F = D!C!BA + D!CBA! + D(C!B! + C) = D!C!BA + D!CBA! + D(B! + C) = D!C!BA + D!CBA! + DB! + DC
F = D!C!BA + D!CBA! + DC + DB! = D!C!BA + C(D!BA! + D) + DB! = D!C!BA + C(BA! + D) + DB!
F = D!C!BA + CBA! + DC + DB!

The Karnaugh map provides a graphical form of simplification. Locate the 1s on the map, form the smallest number of large groups to get the map below. These groups can be read from the map as

F = DB! + DC + CBA! + D!C!BA



A1 Part (d) 6 marks

Design a circuit using NAND logic only to implement the simplified expression for F

F = DB! + DC + CBA! + D!C!BA

We need to convert all OR operators to AND operators.

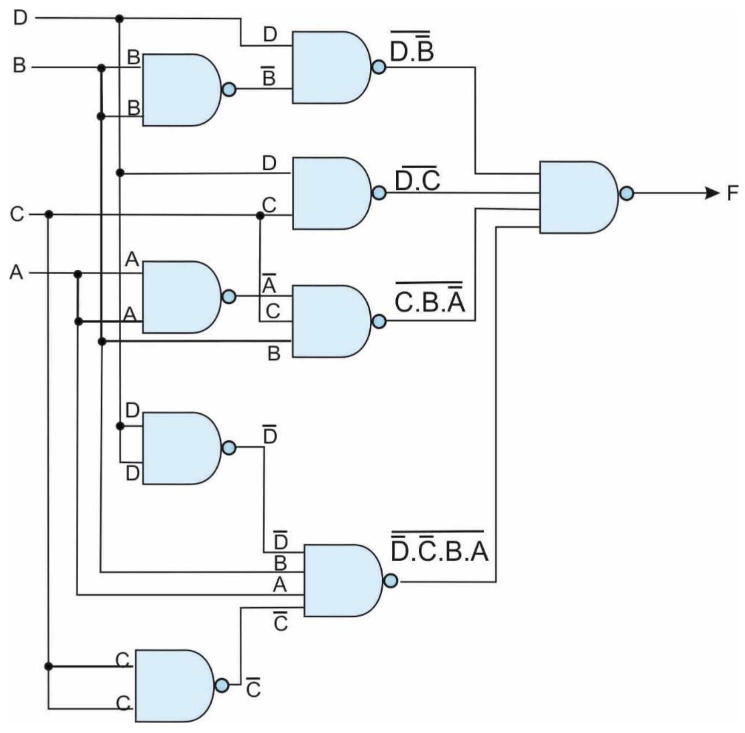
We do this by using DeMorgan’s theorem and noting that complementing (negating) twice does not change a value; that is, F = F

In this case DB! + DC + CBA! + D!C!BA = $\overline{\overline{\text{DB! + DC + CBA! + D!C!BA}}}$

Applying deMorgan’s theorem, we

get $\overline{\overline{\text{F}}\text{ = }\overline{\text{DB!}}\text{ .}\overline{\text{DC}}\text{.}\overline{\text{CBA!}}\text{.}\overline{\text{D!C!BA}}}$

We can now draw a circuit. Note that we can negate a single term by using F = F.F



Examiner’s comments

As it has been noted this question was the most popular with 73.74% of students who attempted it as well as with 81.01% who achieved pass marks.

The structure of the A1 is built in such a way that A1b depends on the answer to A1a, A1c on A1b and A1d on A1c. Thus, it is causing a problem in the case when A1a is answered either partially or mostly incorrectly.

There is evidence that some candidates did not complete the answer A1a correctly, e.g. either just produced a table from 0000 to 1111 without producing output F or have done some portion of the answer or produced the table not in the regular consistent sequence.

When answering A1c, the evidence shows that candidates used Karnaugh maps which was not explicitly required and the provided sample solution was also showing simplification of the formulae.

A1d was attempted by very few students and many of them received low marks. The largest confusion was caused by the requirement to use only NAND logic as some students involved NOT components too. Average mark for the A1 is 18.

A2 Part (a) 7 marks

The performance of microprocessors has been improving exponentially since they were first manufactured. However, over the last few years, the advance of microprocessor performance has slowed. Why is this; what is limiting the increase in microprocessor performance?

There are several reasons why the performance of microprocessors has not increased in accordance with Moore's law over the last few years. The main reason is the ability to increase clock speed.

The speed of a processor is dependent on the rate at which it is clocked. In the 1980s clock rates were in the low megahertz region. Eventually, the 1 GHz clock rate barrier was broken. Today, typical desk-top PC have processors operating in the range of 3 to 4 GHz.

A processor's energy consumption increases with clock speed. Consequently, the processor gets hotter at higher speeds. Since the size of the chip is only a few millimetres square, the heat (which can be over 100 W) can only be dissipated by large heat sinks. The practical limit to clock speed is imposed by the need to cool chips.

Another limitation on processor speed is memory access times. The rate at which processors have increased speed has not been matched by a corresponding increase in memory access times.
Therefore, memory access is proving to be a bottleneck and limits the performance of a computer system.

Examiner's comments

The majority of candidates who answered this question discussed that "microprocessors are getting slower" or wrote general information about the development of computers starting from vacuum tube based machines. Many students stated that "complex applications are causing the slowing down of the microprocessors".

Part (b) 10 marks

How are computer manufacturers attempting to increase the performance of microprocessors?
There are several ways of increasing the performance of a processor. Some of these are:
Pipelining: Instead of completing an instruction before executing the next one, modern processors execute several instructions in a pipeline (like a car assembly line). One instruction is being fetched, while the pervious instruction is being decoded, and the one before that is being executed. A computer with N-stages in the pipeline can provide an N-fold increase in speed with no change in clock rate. However, pipelining cannot provide its full capability because of branches that force the pipeline to be flushed.
Out-of-order execution: The von Neuman machine executes instructions sequentially in order. However, if one instruction is waiting for the result of a computation that is still in the pipeline, instructions further in in the program can be executed if they don't require data that is not yet available.
Multicore: The multicore processor has 2,3,4 or more CPUs on the same chip. Each CPU can process data in parallel. This allows more computations per second by executing them in parallel (but is limited by the ability to divide a program into parallel operations.

Examiner's comments

There is evidence that the majority of candidates who attempted this question repeated part of their response from A2a.

Part (c) 7 marks

Cache memory can increase the performance of a microprocessor system. What is a cache memory and, briefly, explain how it improves performance?

Main memory (usually called DRAM) contains programs and data being executed. Unfortunately, its access time is far too low to be used effectively (typically 50ns access time with a CPU clock of less than 1 ns). Cache memory have a very low access time (e.g., 5ns) and is used to overcome the limitations of DRAM. In general, a program tends to execute the same instructions and data frequently (e.g., 5% of a program may be executed for 90 of the time the program is being executed). Cache memory holds only a fraction of the program and data for the DRAM, but for (typically) 95% of the time data is taken from cache rather than DRAM. The secret of cache is in ensuring that the most frequently used data is stored in cache.

Examiner's comments

There is evidence that the majority of candidates who attempted this question partially repeated what they said in A2b. A small number attempted to draw a diagram showing cache memory, CPU and RAM and indicated use of L1, L2 and L3 cache memories.

Part (d)

The main memory of a computer has an access time of 50 ns. The cache memory has an access time of 5 ns.   The hit ratio for the memory system is 0.9 (i.e., 90% of memory accesses are to the cache). What is the speedup ratio of this system?

The hit ratio is 90%. Therefore, for 0.9 of accesses are to cache and 0.1 of accesses are to DRAM.

The average access times is cache access + DRAM access = 50ns x 0.1 + 5ns x 0.9 = 5 + 4.5 = 9.5ns

The average access time without cache is 1.0 x 50ns = 50ns.

The speedup ratio is access time without cache divide by access time with cache = 50ns/9.5ns = 5.26

Examiner's comments

There is evidence that no candidates answered this question correctly.

A3 Part (a) 5 marks

A computer has a program counter. What is a program counter and what is its function?

The program counter, PC, or instruction counter contains the address of the next instruction to be executed. It is used in a von Neuman computer to step through a program. The instruction at the address in the PC is fetched from memory and executed.

Examiner's comments

There is evidence that candidates struggled with this question as only a few answered correctly.

A3 Part (b) 5 marks

Under what circumstances is the program counter modified by the program?

The program counter is incremented after each instruction is fetched from memory. In this way the program counter steps through the instructions sequentially. However, the program counter can be loaded from memory or a register to execute a jump (or branch) operation. The program counter can be loaded with a new address if a condition is met. For example, BEQ XYZ may mean "branch on condition 0 to address XYZ. This allows the implementation of loops and operations like IF…THEN…ELSE

Examiner's comments

There is evidence that candidates struggled with this question as only a few answered correctly.

A3 Part (c) 10 marks

In the context of computer architecture, what is a stack pointer register? Describe how it is used to implement subroutines in a computer.

The stack pointer register, SP, contains typically the address of the top of stack. A stack is a last-in- first-out, LIFO, queue because items are retrieved from a stack in the reverse order to which they were entered.

In a typical microprocessor stack, an item X is pushed on the stack by
[M[SP]] ← X Push the element pointed on to the stack (in main memory) at the address in the stack pointer
[SP] ← [SP] – 1 Decrement the stack pointer to point to the next free location above the stack

An item is pulled off the stack by
[SP] ← [SP] + 1 Increment the stack pointer to point to the item at the top of the stack
Destination ← [M[SP]] Copy the element at the top of the stack pointer at by SP to its destination

A subroutine is a piece of code that is called, executed and a return is made to the instruction after the calling point. This means that the return address (next instruction after the call) must be preserved.
A typical subroutine call is BSR XYZ which pushed the return address onto the stack and then jumps to location XYZ to execute the subroutine.
At the end of the subroutine an instruction like RTS (return from subroutine) pulls the return address off the stack and puts it in the program counter to execute the instruction after the return. This mechanism allows subroutines to be next (one subroutine to call another).

Examiner's comments

There is evidence that candidates struggled with this question as only a few answered with sufficient level of details (e.g. diagram with stack, etc.).

A3 Part (d) 10 marks

What is the function of an index register (also called pointer register or an address register) in a CPU. Give a simple example of the use of an index register.

An index register is a pointer register. That is, it contains the address of an operand If register A0 is an index register then an instruction of the typical form MOVE D0,[A0] means “copy the item pointed at by the contents of index (pointer) register A0 into data register D0”.

The index register allows you to use variable addresses because you can change an index register’s contents. This allows you to implement data structures such as lists, vectors, tables, and arrays.

A typical example of the use of indexing might be:

```
    MOVE A0, Table1 Index register A0 points to Table1 (source)
    MOVE A1, Table2 Index register A1 points to Table2
    (destination)
Loop MOVE D0,[A0]   Repeat: read item pointed at by  A0
    ADD A0,#1            increment A0 to point to next item
    MOVE [A1],D0           copy item to location pointed at by
    A1 ADD A1,#1         increment A1 to point to next  item
    UNTIL Done     EndRepeat … suitable code to close the loop Note … any suitable example
```

of indexing is acceptable.

Examiner’s comments

There is evidence that candidates struggled with this question as only a few answered with sufficient level of details (e.g. ASSEMBLER-like code or diagram with references to the parts of memory, etc.).

A4 Part (a) 10 marks

The operation of the simple von Neumann computer with its fetch/execute cycle has been enhanced by several techniques to improve its performance. Write notes on the following three mechanisms that are found in most computers. In each case, state what the technique is, how it works, and why it increases computer performance. THE INTERRUPT

A computer executes a program instruction by instruction until the program has been completed. However, there are occasions when a computer needs to interact with its environment. Without an interrupt mechanism, the computer would have to periodically text environment variables (e.g., the time, input and output device status, the readiness of disk drives and printers etc.).

An interrupt provides a mechanism whereby any device wanting attention (e.g., a printer, keyboard, or a mouse movement) can inform the computer that attention is required on a hardware line connected to the computer called an interrupt request, IRQ.

Then an interrupt request is received, the computer determines whether the level of the request is high enough to be serviced. If it is, the current operation is completed, the return address saved and a jump made to the appropriate interrupt handler (the software that deals with the source of the interrupt; for example, a printer driver). When the interrupt has been serviced, the return address is loaded into the program counter and execution continues.

Some interrupts (often called exceptions) are triggered by software events such as an illegal operation (invalid instruction) or a memory error (page fault). In this case, the operating system usually deals with the cause of the interrupt.

Interrupts make a computer more efficient and responsive. However, they also make its behaviour harder to predict because interrupts are usually asynchronous and can happen at any time.

Examiner’s comments

There is evidence that a small number of candidates answered this question. Many discussed “microprocessors and their architecture” or wrote general information about operating systems.

A4 Part (b) 10 marks

VIRTUAL MEMORY

In the simple model of a computer, the CPU generates an address and that is used to access data or an instruction in memory.

In reality, the simple model of CPU and memory is not appropriate for several reasons. First, the actual memory and the memory accessed by the computer may be different in size. This was a major problem in the past because many programs were bigger than the available DRAM storage and could not fit into memory.

A second problem is that memory must be segregated and allocated individual users and the operating system. This is necessary for security. Much of the malware found on computers operates by bypassing security mechanisms and allowing the rogue software to access data that should have been hidden/protected.

A third problem is that the programmer should not have to worry about memory addresses and where data goes in memory.

A virtual memory system uses a memory management unit, MMU, that sits between the address bus from the CPU (the virtual address) and the DRAM memory (the physical address). The MMU translates virtual addresses into physical addresses (location of the actual data).

The MMU is able to check the “rights” or “permissions” of every virtual address generated by the CPU. If the address is accessing a region of memory that has been allocated to it, the address is translated into the appropriate physical address. If not, the access is terminated.

Virtual memory allows programs to reside on the hard disk. When data or program is accessed and it not currently in memory, the operating system intervenes and moves it from disk to DRAM and automatically updates its logical-to-physical address mapping tables.

Examiner’s comments

There is evidence that a small number of candidates answered this question. Many discussed “virtual memory” while they actually were referring to “cloud computing”.

A4 Part (c) 10 marks

DIRECT MEMORY ACCESS (DMA)

Memory is normally accessed by the processor … instruction codes when executing a program and data when processing information. When data is required from the outside world (e.g., data from a disk drive or from the Internet), it must be read into memory.

In a simple computer, the CPU may be programmed to read data and put it in memory (input) or to read memory and transfer it to a peripheral (output). This process is slow and inefficient and suitable only for simple controller applications.
Direct Memory Access, DMA, is a mechanism whereby the computer’s data and address buses are disconnected from the CPU (electronically) and a DMA controller takes them over. The DMA controller is an autonomous device (a little like a CPU in its own right). The DMA controller is able to control the buses to transfer data between a peripheral and memory rapidly and efficiently.
The DMA controller can be arranged to perform I/O data transfers in periods when the CPU is not using the memory’s bus or it can interleave data operations with the CPU (so called cycle stealing). In a modern computer like the PC, the DMA function is built into the “bridge” controller chips.
Essentially, any card in a PCI slot can request the bus from the computer, take over the bus (i.e., become the bus master) and then perform the DMI.

Examiner’s comments

Very few candidates answered this question (much less than A4a and A4b) and even less followed it to the point. None of the candidates referred to the PCI cards and their operation. Candidates should cover all topics in the syllabus.