Examiners' commentary 2018–2019

CO1110 Introduction to computing and the internet – Zone B

General remarks

The examination was set to test the candidate's basic and deeper understanding of the material contained in the subject guide, Introduction to computing and the internet. The examination is split into two parts, A and B. Candidates had to answer two questions from each part.

Comments on specific questions

Question 1

a. Answer

i. (C) Fixed point notation

ii. (C) When the positive exponent is too large to be expressed in the number of bits available

iii. (D) All of the above

Comments

Only a very small minority gave completely correct answers to part (a). About 80 per cent of candidates gave the wrong answer to (i), with D: *None of the above* the most popular incorrect answer. The correct answer, fixed point notation, is discussed in section 6.4.1 of Volume 1 of the subject guide.

Less than half of candidates gave the correct answer to (ii), with A: *When the mantissa is too large to be expressed in the number of bits available* being by far the most popular incorrect answer. Candidates should understand that when the mantissa is too large to be expressed in the number of bits available, then it is rounded, with some loss of precision. The problem is the exponent, because it has to be exact, so no loss of precision is possible.

Nearly all candidates gave a wrong answer to part (iii). These candidates failed to appreciate that while the statement that they gave was true, all statements (A) to (C) were true, making (D) the correct answer.

b. Answer

i.
```
7            -7
0111         1001
```

ii.
```
1011
1010
-----
10101
```

Answer after discarding the extra bit: `0101`

The leading zero in the result indicates both a positive number, and a change of sign from the operands, which are both negative numbers since they both have 1 as their sign bit. The change of sign bit indicates overflow.

iii.
```
111011
111010
------
1110101
```

Answer after discarding the extra bit: 110101

There is no difference in the sign of the operands and the result; hence there is no overflow.

Comments

Part (i) was normally answered correctly, with most candidates understanding that in order to find the positive representation of a two's complement number, one first has to find the unsigned binary representation of the number, which for 7 is 111. A leading zero is then added to indicate a positive number in two's complement, hence 7 is 0111 in two's complement. To find a negative number, first find its positive two's complement representation, then use one of the methods described in section 6.3.8 of Volume 1 of the subject guide to transform this into the negative representation of the number. Hence to find –7 transform the two's complement representation of 7, which is 0111, to its negative two's complement counterpart, 1001.

Part (ii) was also usually answered correctly, with a few candidates who did not discard the extra bit, but instead incorrectly claimed that there was overflow because adding up 4-digit numbers had resulted in a 5-digit number. As the subject guide explains in section 6.3.8, when addition in two's complement results in an extra bit, that bit is discarded to give the correct answer.

In part (iii) almost half of candidates received no credit, as their 6-bit versions of the 4-bit two's complement numbers A = 1011 and B = 1010 were incorrect. Often these candidates simply filled the extra bits with leading zeros, for example transforming A into 001011. These candidates should have known that this was incorrect, as it had the effect of turning 1011, a number with a negative sign bit, into 001011, a number with a positive sign bit, hence it is not possible that the two numbers are the same. Other candidates added zeros after the sign bit to make a 6-digit number, for example transforming 1011 into 100011, which at least gave a negative number, but was still incorrect.

As the subject guide explains in Volume 1, section 6.3.13, to convert a two's complement number to a two's complement number with more bits, it is first necessary to move the sign bit to the leftmost position in the new number, and then fill in the gaps with copies of the sign bit. Hence the first stage in transforming A to a two's complement number with 6-bits would be 1 _ _ 011. The spaces are then filled with copies of the sign bit, giving 111011.

c. Answer

i. 31 is positive so the sign bit is zero

$31_{10} = 11111_2$, hence the mantissa is = 1.1111×2^4

The leading 1 is implicit, so the normalised mantissa is

1111 0000 0000 0000 0000 000

The exponent of the mantissa is 4. Biasing it by 127 gives 127 + 4 = 131

$131_{10} = 1000\ 0011_2$

Hence 31 in normalised IEEE 754 single precision format is:

0 1000 0011 1111 0000 0000 0000 0000 000

ii. $0.001101111 \times 2^{-126}$

Comments

Part (i) was answered well on the whole, with most candidates knowing how to make the transformation. The most common mistake seen was candidates who gave the mantissa as 0.11111 x 25, which meant that their exponent was incorrect. Another mistake sometimes seen was when candidates subtracted 127 to find the biased exponent, rather than adding it.

In part (ii) about a third of candidates had no idea how to answer the question, which is covered in section 6.4.8 of Volume 1 of the subject guide. Another third gave either the correct answer, or gave 0.01101111 x 2–127 as their answer, not understanding, or forgetting, that the –127 exponent is reserved for special values and not otherwise used. Another minor mistake was in applying rounding to the mantissa, i.e. 0.00111 x 2–126, when such loss of precision is not necessary.

Another third of candidates transformed their answer into IEEE 754 denormalised representation. These candidates did not lose any marks, but did much unnecessary work when the above answer was all that was required. The examiners had tried to indicate this by the wording of the question, which stated that the number given should be converted 'to a form that ***can be expressed*** in IEEE 754 32-bit denormalised form'. The question also stated that candidates could give the exponent in decimal notation, and this was another indication that the above answer was all that was necessary.

Question 2

a. Answer

i. (A) An opcode

ii. (C) Program counter

iii. (A) Floating point operations per second

Comments

The majority of candidates gave correct answers to part (a). Part (ii) was most likely to be answered incorrectly with nearly half of candidates giving a wrong answer, fairly evenly divided between A: *Instruction register* or B: *Memory address register*.

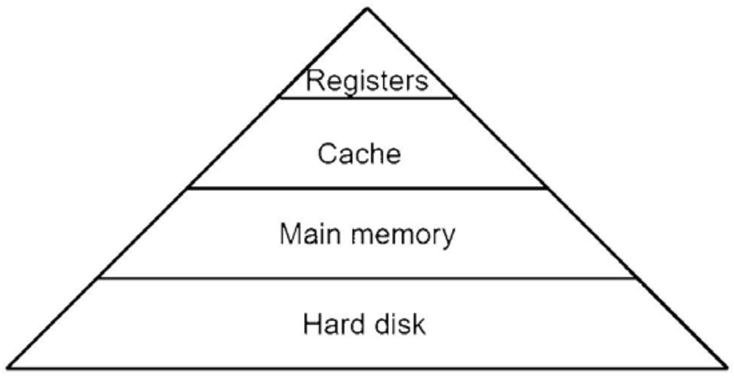
b. Answer

i.

- **Storage grows** with increasing distance from the processor

  + one of the following:
- **Cost of memory** shrinks with increasing distance from the processor
- **Speed of memory** shrinks with increasing distance from the processor
- **Frequency of access** decreases with increasing distance from the processor

ii.



iii. With pipelining, M instructions can take **N + M – 1** time units.

Comments

In part (i) almost all candidates correctly answered that storage (or size of memory) grows with increasing distance from the processor, while a few candidates gave an incorrect answer to the second part of the question, which asked for one thing that decreased with increasing distance from the processor. One popular incorrect answer was that the size of the cache shrunk with increasing distance from the processor. These students were presumably referring to multilevel caches, i.e. where the processor is supported by more than one cache. While this is not covered in the subject guide, generally the L1 cache will be smaller and faster than the L2 cache.

About 20 per centof candidates gave an answer to part (ii) that was completely incorrect. There was also a small number of candidates who gave the correct hierarchy but in reverse order, with the *Hard disk* at the top of the pyramid, and *Registers* at the bottom. Despite this, most candidates answered correctly.

About 40 per cent of candidates gave an incorrect answer to part (iii), with NM/2 the most popular wrong answer, with M/N and M/N –1 also seen. For a pipeline that works perfectly N + M –1 is the fastest possible time to execute M instructions. Some candidates wrote comments with their answer that N + M –1 was an ideal unlikely to be reached in practice, which is correct.

With pipelining, once the first instruction has started, in the next cycle the second instruction can be started, and in the next the third. Hence instructions can overlap each other, meaning that the time taken for M instructions in a pipeline that operates perfectly will be the time for one instruction to complete (which is N), plus an extra one for each instruction apart from the first, hence N + M –1. For example, if an instruction has five stages, then three instructions could take 5 + 3 –1 = 7 cycles (or time units) to complete. This can be seen in the table below, where the instruction cycle consists of five stages IF; ID; EX; MEM; and WB:

Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Instruction 1	IF	ID	EX	MEM	WB		
Instruction 2		IF	ID	EX	MEM	WB	
Instruction 3			IF	ID	EX	MEM	WB

c. Answer

i. Spatial locality states that memory locations accessed in any time period are likely to be clustered in one place.

because instructions are normally accessed sequentially.

To exploit this tendency, when the cache admits a new memory unit, it also admits the surrounding memory units.

It does this by dividing memory into blocks, each block fits into a line in the cache. When a new memory unit is required, the cache admits the entire block containing that memory unit into a line in the cache.

ii.

Main memory block numbers		**Cache lines**
{0, 4, 8, 12, 16}	➔	Line 0
{1, 5, 9, 13, 17}	➔	Line 1
{2, 6, 10, 14, 18}	➔	Line 2
{3, 7, 11, 15, 19}	➔	Line 3

iii. (A) Direct mapped cache address
(B) Associative mapped cache address
(C) Set associative mapped cache address

Comments

Less than 15 per cent of candidates gave an answer to part (i) that covered all of the necessary points for full credit. Candidates needed to define spatial locality clearly, which many candidates could not do, with many writing that the cache stores recently used items as they are likely to be used again. In fact, the cache uses the spatial locality principle to *predict* that items near to a requested memory unit are likely to be needed next, or in a short period of time. Many candidates stated that when the cache admits a new memory unit, it also admits surrounding, or nearby, memory units, but could not, or did not, state how this is implemented by dividing memory into blocks that fit in a cache line.

In part (ii) just less than half of candidates gave a correct answer. Some candidates had the right idea, but their execution was off by one, meaning that block numbers for cache line 1 were assigned to line 0, those for cache line 2 assigned to line 1, and so on. Many candidates had no idea, writing for example that memory block numbers 0–4, or 1–5 would map to line 0, 5–9 or 6–10 would map to line 1, and so on.

In fact, with 4 cache lines block numbers need to be divided up according to their remainder with respect to division by 4. Those that have a zero remainder (0 and numbers divided by 4) will map to line 0, those that have remainder 1 when divided by 4 (1, 5, etc.) will go into line 1, those with remainder 2 into line 2 and those with remainder 3 into line 3. The organisation of direct mapped caches is discussed in Volume 1 of the subject guide, section 4.7.4.

Most candidates gave the correct answers to part (iii), although a few candidates confused *direct mapped* with *associative mapped*, and vice versa.

Question 3

a. **Answer**

i. (B) CPU, memory and I/O devices
ii. (D) All of the above
iii. (A) Integrated circuits

Comments

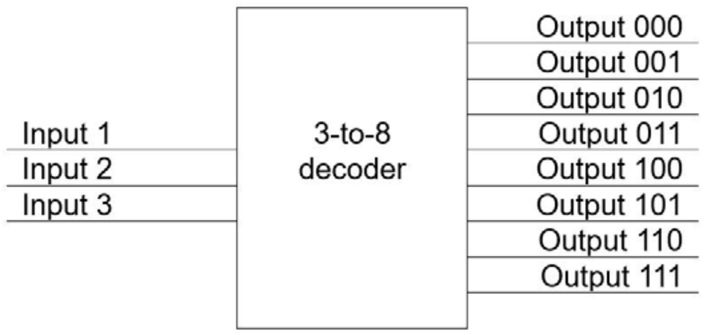
Only a few candidates gave completely correct answers to part (a). About two thirds of candidates gave an incorrect answer to (i), with C: *Control unit, address decoder and system bus* the most popular incorrect answer, followed by D: *None of the above*.

Part (ii) was most likely to be answered correctly, with just over one third of candidates giving incorrect answers, usually (A) or (C). These candidates did not understand that all of the statements (A), (B) and (C) were correct, making (D) the correct answer.

Most candidates gave an incorrect answer to (iii), with C: *Logic gates* by far the most popular wrong answer.

b. Answer

i.



ii. Each chip has 28 bytes on it. There are 32, or 25 chips. Since memory is byte addressable, the number of addressable memory locations is the number of bytes. There are 28 x 25 = 213 bytes, so the number of addressable memory locations is 213.

This means that 13 bits are needed for memory addresses.

iii. There are 25 chips, so 5 address lines for chip select.

For byte selection, 13–5 = 8 address lines are needed.

Comments

Most candidates answered part (i) correctly, although for some reason a small number gave the output numbers as 4 digit binary numbers by adding an extra zero to the leftmost position.

In part (ii) only a few correct answers were seen. The only common error was committed by those candidates who treated the memory as bit addressable, rather than byte addressable, and hence gave their answer as 16 rather than 13 (since if there are 213 bytes there are 2^{13} x 8 bits = 2^{13} x 2^{3} = 2^{16}). This is discussed further in section 3.5.2 of Volume 1 of the subject guide.

Few candidates received full credit for part (iii) of the question, as completely correct answers were rare. Candidates who had incorrectly calculated the number of bits needed for memory addresses in part (ii) as X, and gave their answer to the number of lines needed for byte selection as X–5, received some credit for using the correct method, i.e. number of bits needed for a memory address minus the number of bits needed for chip select (5 in this memory).

c. Answer

i. **Programmed I/O:** The CPU issues a read/write command to the device's I/O module and sits idle. The CPU wastes time in checking if the state for the I/O module of the device is ready.

Interrupt-Driven I/O: The CPU issues a read/write command to the device's I/O module then goes to do other things. When the I/O module is ready it uses an interrupt signal to tell the CPU it is ready to read/write from/to the device.

Unlike Programmed I/O, the CPU does not waste time in waiting and checking if the I/O module of the device is ready.

ii. In both programmed and interrupt-driven I/O the CPU spends a lot of time transferring data as it has to handle the transmission of each word of data.

Comments

Almost one quarter of candidates attempting this question gave no answer to part (c). Those that attempted an answer to (i) were often clearly guessing at what programmed I/O and interrupt-driven I/O were, based on their names, for example 'programmed I/O is when I/O is programmed to be done in a certain order...'. Most of these candidates demonstrated no understanding that both programmed I/O and interrupt-driven I/O are different I/O protocols that the processor could follow. Section 5.8 of Volume 1 of the subject guide discusses programmed I/O and interrupt-driven I/O further.

Where candidates did understand what programmed I/O and interrupt-driven I/O were, they often lost marks by being vague, many wrote that with programmed I/O the processor issues a command and then waits, but did not state that while it is waiting the CPU checks (or polls) I/O devices for readiness. Quite often the comparison between the two I/O protocols was also not clearly and explicitly made.

While few candidates received full marks for part (i), even fewer received full credit for (ii), with most candidates either having no idea and clearly guessing, or losing some credit by giving vague answers such as 'the processor wastes time in both schemes'.

Question 4

a. Answer

i.	(A)	Network Access → Internet → Transport → Application
ii.	(B)	TCP
iii.	(D)	All of the above

Comments

Most candidates gave completely correct answers to part (a) with only a very small number of wrong answers seen.

b. Answer

i. When the time-to-live field reaches zero, the datagram is discarded and an ICMP message sent to the source host.

ii. The protocol field records the Transport Layer protocol that will interpret the data.

iii. The checksum verifies that the **header** has not been corrupted.

Comments

Most candidates answered (i) partly correctly, stating that the datagram would be discarded when the time-to-live reached zero, but not stating that an ICMP message would be sent to the message source. Very few candidates answered part (ii) correctly, with a few stating that the field specified the protocol that would interpret the data without saying that this would be a transport layer protocol, although some implied this by stating that the protocol would be TCP or UDP. Most answers to (iii) were incorrect, while some candidates lost some credit by giving answers that were correct but not precise enough, for example 'the checksum verifies the data'.

c. Answer

i. For full credit an answer should have included the following two points:

- Data packets are expected to be received in order, and only data received in order will be acknowledged.
- An acknowledgement confirms receipt of all unacknowledged data received with a smaller sequence number.

ii. Assume a situation where the receiver has received packet x–1, followed by packets x+1, x+2, followed by many other in-order packets. The receiver cannot signal to the sender that while it has not received packet x, it has received packet x+1, followed by a great many other in-order packets.

- If the sender follows the accepted standard and retransmits only the first unacknowledged segment, it must wait for the acknowledgement before it can decide what and how much to resend. Thus, retransmission reverts to a send-and-wait paradigm, slowing overall transmission time.
- Alternatively, the sender can send again all packets starting with packet x, even though many of them have already been successfully received. This is a potentially large overhead.

Comments

In answer to part (i) some candidates gave a description of the 3-way handshake, used to establish a TCP connection. This is not the same as cumulative acknowledgement, which is about what happens after the connection has been established. For full credit candidates needed to give an answer that contained the two numbered points given above, which very few did. Some candidates stated that the receiver will acknowledge all packets, or stated that each packet that is sent is acknowledged, without giving any more detail and hence receiving no credit. Others stated that the receiver acknowledges each packet with its sequence number, which again was too general and lacking in detail to receive any credit.

In answer to part (ii) most candidates only made the point that where packets were received out of order, the missing packet may be sent again, together with all other packets sent from that point, a potentially large overhead. These candidates received most of the credit for part (ii), but for full credit should also have covered the first bullet point above. Only a very small minority of candidates included the first bullet point above in their answer.

Question 5

a. Answer

i.	(A)	IMAP
ii.	(B)	Inline; Document level; External style sheet

OR

	(C)	External style sheet; Document level; Inline
iii.	(C)	201.168.67.0/27

Comments

Mostly correct answers to part (a) were seen, even though almost half of candidates gave an incorrect answer to (i), with D: *None of the above* the most popular incorrect answer.

In part (ii) most candidates answered (B), with a minority answering (C). Since the question implied but did not state clearly that the order of precedence should be from highest to lowest, the examiners decided to accept both (B) and (C) as correct.

Most candidates answered (iii) correctly, with about one quarter giving an incorrect answer, usually D: *None of the above*. CIDR notation consists of the network address, followed by a decimal number indicating the number of leading 1s in the subnet mask when given in binary octets. The subnet mask is 255.255.255.224, which is 11111111 11111111 11111111 11100000 in binary octets, hence there are 27 1s. The number 27 indicates the number of bits that are to be considered as part of the network address, with the rest of the bits in the network address borrowed for use in assigning subnets.

b. Answer

i. The subnet mask is 255.255.255.248

= 11111111 11111111 11111111 11111000.

Number of bits borrowed from the network address is 5.

Hence the number of possible subnets is $2^5 = 32$

(accepted $2^5 - 2 = 30$).

ii. The number of bits representing the host is 3, hence the number of possible hosts in each subnet is $2^3 - 2 = 6$.

iii. The address of the first subnet is 223.132.129.0

Host addresses: 223.132.129.1 – 223.132.129.6

Also accepted: first usable subnet 223.132.129.8

Host addresses: 223.132.129.9 – 223.132.129.14

Comments

About a quarter of students got this question completely wrong and scored nothing.

In considering answers to part (b) the examiners were aware that the current subject guide states that out of every set of possible subnets, two will be unusable for technical reasons. Since the guide was published the situation has changed, hardware has been designed to cope with all subnet addresses. Because of this the examiners decided to accept either 2^5 or $2^5–2$ as the correct answer to (i), and to accept the address of the first subnet in part (iii) as 223.132.129.**0 or** 223.132.129.**8.** Similarly, there are two possible ranges of host addresses; both were accepted as correct.

Of those candidates giving a correct answer to (iii) about half gave 223.132.129.0 and half gave 223.132.129.8. Those giving 223.132.129.0 tended to do so with a comment that they were assuming that all subnet addresses were usable.

Most candidates answered part (ii) correctly, while a few lost credit by answering $2^3 = 8$. While it is the case that the all zeros and the all 1s subnet addresses can now be used, it is not the case that the all zeros and all 1s host addresses can be used, hence 2 must be subtracted from the potential host addresses to give all possible host addresses.

c. Answer

i. (A) External style sheet

(B) Document level

(C) Inline

ii. For full credit an answer should have included the following three points:

CSS separates style and content, with the style described by CSS or equivalent, e.g. one department can specialise in design, while other departments focus on their web content.

A CSS file, with the house style, can be linked to by all persons and departments developing and maintaining web pages for the organisation. Hence the work of specifying the style only needs to be done once, rather than separately for each web page.

If changes to the house style are needed, then instead of changing individually possibly hundreds of web pages, only the content of the CSS file needs to be changed.

Comments

Answers given to (i) were usually correct, and all answers to (ii) gained some credit. Many candidates lost credit in part (ii) by giving answers that were too short and not detailed enough; for example, many did not explicitly say that CSS allows the separation of style and content.

Question 6

a. Answer

i.	(A)	Malicious code that pretends to be a legitimate program, such as a helpful utility, or a game
ii.	(D)	All of the above
iii.	(A)	A patent must be applied for

Comments

Mostly correct answers to part (a) were seen, even though nearly half of candidates gave an incorrect answer to (ii), failing to appreciate that all of the statements (A), (B) and (C) were correct.

b. Answer

i. DoS attacks seek to put a computer system out of action, or to slow it down such that legitimate users find it hard to access, by overwhelming it with network requests. It does this by exploiting features of legitimate protocols in order to tie up server resources.

ii. A number of computers with the same RAT installed may be networked, so that the hacker can send one command that all machines will carry out. This can mean a DDoS attack from potentially millions of devices can be launched with one command.

iii. A DoS or DDoS attack is hard to defend against because it exploits features of legitimate protocols. Since the network requests are legitimate, it is very hard to distinguish between requests from genuine users, and those with a malicious intent.

Comments

Part (i) was normally answered correctly, although often answers were vague and did not, for example, note that network requests in a DoS attack were usually legitimate.

Most answers to part (ii) were wrong, with the most common incorrect answer being that a DDoS attack could be used as a distraction in order to implant a RAT. To receive any credit candidates needed to at least explain that many machines infected with the same RAT could be used to launch a DDoS attack.

In part (iii) many candidates considered that the reason that DoS attacks are hard to defend against is that it is impossible to predict when an attack may happen, or that it is because it is hard to know where the attack originates from. In fact, the examiners were looking for candidates to answer that a Dos or DDoS attack is hard to defend against because the network requests are normally legitimate, hence it is hard to tell which requests are from legitimate users, and which are malicious. Candidates were given some credit for discussing protection and mitigation of DoS attacks; for example, some candidates made the points that: : (1) anti-virus software and firewalls may not be effective against DoS attacks; (2) one defence can be to block or divert requests, but this also affects legitimate users; and (3) another defence is to increase the bandwidth of the server.

c. Answer

There is no right or wrong answer to this question. A good answer could argue for or against software patenting, providing that the arguments made were sound. The examiners were looking for answers that made sensible, relevant points, deployed coherent argumentation, showed depth of knowledge and understanding, were well-structured and written with clarity.

Comments

Candidates lost marks for this question by putting forward arguments that were undeveloped and lacking in consideration of counter arguments. For example, many candidates argued that patenting is necessary, as it means that developers can profit from their software, without seeming to be aware that many companies sell and profit from software that is not patented, and cannot be patented in most jurisdictions.

Many candidates wrote about the practice of patent trolling, but were not clear that this is mainly a problem in the USA. These candidates also did not explore whether allowing the patenting of software would inevitably encourage patent trolling, or whether patent trolling was a peculiarly American phenomenon. Many candidates made the point that patents guarantee the right to exploit an invention, which in turn safeguarded the income of inventors. Other candidates explored more nuanced arguments to do with the expense of getting and enforcing a patent, meaning that often patent protection is only available to larger companies, and that smaller businesses may be at a disadvantage, or may even be pushed out of the market altogether.