

CAMBRIDGE INTERNATIONAL EXAMINATIONS**Cambridge International Advanced Level****MARK SCHEME for the October/November 2015 series****9608 COMPUTER SCIENCE****9608/32**

Paper 3 (Written Paper), maximum raw mark 75

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the October/November 2015 series for most Cambridge IGCSE[®], Cambridge International A and AS Level components and some Cambridge O Level components.

® IGCSE is the registered trademark of Cambridge International Examinations.

Page 2	Mark Scheme	Syllabus	Paper
	Cambridge International A Level – October/November 2015	9608	32

- 1 (a) (i) 01101000 0011
 $= 0.1101$ (or $\frac{1}{2} + \frac{1}{4} + \frac{1}{16}$) $\times 2^3$ [1+1]
 $= 110.1$
 $= 6.5$ [1]
- (ii) +3.5
 $= 11.1$ [1]
 $= 0.111 \times 2^3$ (or indication of moving binary point correctly) [1]
 $= 01110000 0010$ [1]
- (iii) 01110000 Allow f.t. from (ii)
10001111 One's complement on mantissa [1]
10001111 +1 Two's complement [1]
 $= 10010000 0010$ [1]
- (b) (i) Precision/accuracy of numbers represented will increase [1]
(ii) Range of numbers represented will increase [1]
- (c) Any point, 1 mark (max. 3)
- 0.1/0.2 cannot be represented exactly in binary // rounding error [1]
0.1 represented by a value just greater than 0.1 // 0.2 represented by a value just greater than 0.2 [1]
adding two representations together adds the two differences [1]
summed difference significant enough to be seen [1]
[max. 3]

[Total: 14]

2 (a)

Symbol	Token	
	Value	Type
Start	60	Variable
0.1	61	Constant
Counter	62	Variable
10	63	Constant

[1]

[1+1]

Page 3	Mark Scheme	Syllabus	Paper
	Cambridge International A Level – October/November 2015	9608	32

(b)

60	01	61	4E	62	01	60	50	63	52	62	02	60	53
----	----	----	----	----	----	----	----	----	----	----	----	----	----

[1+1]

(c) (i) syntax analysis

[1]

(ii) any **two** points from:

construct parse tree // parsing
 checking syntax/grammar
 produce error report

[max. 2]

(d) (i) Minimise the execution time // code runs faster

[1]

(ii) Compiler could calculate $2*6$ and replace it with the value 12.

[1]

(iii) LDD 436
 ADD 437
 STO 612
 ADD 438
 STO 613

}
 } [1]
 }
 [1]
 [1]

-1 for each additional instruction; 0 for copy of original code

[Total: 13]

3 (a) dedicated circuit/channel/physical path
 which lasts for duration of connection

[1]
 [1]

(b) e.g.

cs: gives dedicated circuit
 ps: split into packets/chunks
 ps: sends packets on individual routes
 cs: whole bandwidth available // ps: shares bandwidth
 cs: faster data transfer
 cs: packets arrive in order they are sent
 cs: packets cannot get lost
 cs: better for a real-time application
 ps: packets may arrive out of order so delay until packet order restored
 ps: packets may get lost so retransmission causes delays

[1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [max. 6]

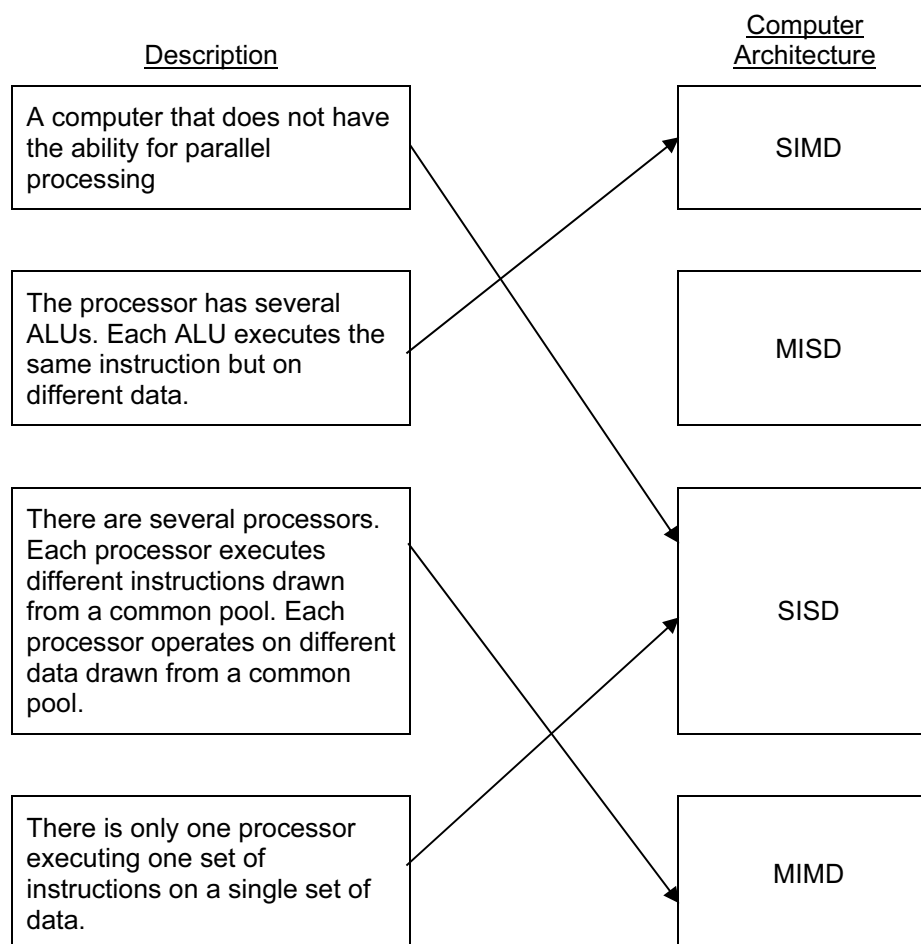
(c) web page divided into packets/chunks
 each packet has destination address
 router looks at IP address...
 and decides where to send packet next for most efficient path
 packets can take different routes
 home computer reassembles packets to rebuild web page

[1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [max. 3]

Page 4	Mark Scheme	Syllabus	Paper
	Cambridge International A Level – October/November 2015	9608	32

[Total: 11]

4 (a) 1 mark for correct arrow from each description



[4]

(b) (i) **Massive:** many/large number of processors // hundreds/thousands of processors [1](ii) **Parallel:** to perform a set of coordinated computations in parallel/simultaneously [1]

(c) processors need to be able to communicate ... [1]

so that processed data can be transferred from one processor to another [1]

suitable algorithm/program/software/design // appropriate programming language [1]

which allows data to be processed by multiple processors simultaneously [1]

[Total: 10]

Page 5	Mark Scheme	Syllabus	Paper
	Cambridge International A Level – October/November 2015	9608	32

5 (a) (i)

$$Z = P \cdot \overline{Q} \cdot \overline{R} + P \cdot \overline{Q} \cdot R + P \cdot Q \cdot R$$

[1]
[1]
[1]

(ii)

		PQ			
		00	01	11	10
R	0	0	0	0	1
	1	0	0	1	1

[1]

(iii) 1 mark each loop

		PQ			
		00	01	11	10
R	0	0	0	0	1
	1	0	0	1	1

Allow f.t. from (ii)

[2]

(iv)

$$Z = P \cdot \overline{Q} + P \cdot R$$

[1]
[1]

Allow f.t. from (iii)

(b) (i) 1 mark row headings. 1 mark column headings.
1 mark per 2 correct rows (based on headings)

		PQ			
		00	01	11	10
RS	00	0	0	0	0
	01	0	1	1	1
	11	0	1	1	0
	10	0	0	0	0

[4]

Page 6	Mark Scheme	Syllabus	Paper
	Cambridge International A Level – October/November 2015	9608	32

- (ii) 1 mark for loop with two 1s; 1 mark for loop with four 1s

PQ

		00	01	11	10
RS	00	0	0	0	0
	01	0	1	1	1
	11	0	1	1	0
	10	0	0	0	0

Allow f.t. from (i)

–1 for each incorrect grouping, max. 2 errors

[2]

- (iii)

$Z =$

$Q.S$

[1]

$+P.R.\bar{S}$

[1]

Allow f.t. from (ii). –1 error if more than 2 terms

[Total: 16]

6 (a) blocked → ready:

process is waiting for resource/I/O operation to complete (blocked state)

[1]

when I/O operation completed process goes into ready queue (ready state)

[1]

running → ready:

when process is executing it is allocated a time slice (running state) // process is allocated time on processor

[1]

when time slice completed/interrupt occurs process can no longer use processor even though it is capable of further processing (ready state)

[1]

(b) to be in blocked state process must initiate some I/O operation

[1]

to initiate operation process must be executing

[1]

if process in ready state cannot be executing/must be in running state

[1]

(c) (i) exit/termination/completion

[1]

(ii) when the process has finished execution

[1]

(d) low-level scheduler:

decides which of the processes in ready state

[1]

should get use of processor/be put in running state

[1]

based on position/priority

[1]

invoked after interrupt/OS call

[1]

[max. 2]

[Total: 11]