

Please write clearly in	n block capitals.
Centre number	Candidate number
Surname	
Forename(s)	
Candidate signature	I declare this is my own work.

AS COMPUTER SCIENCE

Paper 2

Tuesday 21 May 2024

Afternoon

Time allowed: 1 hour 30 minutes

Materials

For this paper you must have:

· a calculator.

Instructions

- Use black ink or black ball-point pen.
- Fill in the boxes at the top of this page.
- Answer all questions.
- You must answer the questions in the spaces provided. Do not write outside the box around each page or on blank pages.
- If you need extra space for your answer(s), use the lined pages at the end of this book. Write the question number against your answer(s).
- Do all rough work in this book. Cross through any work you do not want to be marked.

Information

- The marks for questions are shown in brackets.
- The maximum mark for this paper is 75.

Advice

- In some questions you are required to indicate your answer by completely shading a lozenge alongside the appropriate answer as shown.
- If you want to change your answer you must cross out your original answer as shown.
- If you wish to return to an answer previously crossed out, ring the answer you now wish to select as shown.

For Exam	iner's Use
Question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
TOTAL	_

Answer all questions in the spaces provided.						Do out
0 1 . 1	Describe the se	et of rea	al numbers.		[1 mark]	
1.2	The number 5	can be	written as $\frac{15}{3}$			
	Shade two loze	enges t	to indicate which of the following stateme	ents are true	e. [2 marks]	
		Α	15 and 3 are not integers	0		
		В	15 and 3 are irrational numbers	0		
		С	5 is an irrational number	0		
		D	5 is a natural number	0		
		E	5 is a rational number	0		
1.3	Shade one loz		o indicate which of the symbols below rep	presents the		
	rational numbe				[1 mark]	
	rational numbe	A	C	0	[1 mark]	
			C N	0	[1 mark]	
		Α			[1 mark]	
		A B	N	0	[1 mark]	



0 2.1	Convert the bit pattern 10001010 to hexadecimal.	[1 mark]	Do not write outside the box
0 2.2	Represent the decimal number 139 as an 8-bit unsigned binary integer .	[1 mark]	
0 2.3	Show how the unsigned binary number 00100011 can be added to the binary number 00101011 without converting the numbers into decimal. You must show all your working in binary.	unsigned [2 marks]	
	0 0 1 0 0 0 1 1 + 0 0 1 0 1 0 1 1		
	Question 2 continues on the next page		



		1
0 2.4	Show how the 8-bit two's complement binary integer 00011100 can be subtracted from the 8-bit two's complement binary integer 00111011 without converting the numbers to decimal.	Do not writ outside the box
	You must show all your working in binary.	
	[2 marks]	
0 2 . 5	The bit pattern in Figure 1 represents a 10-bit unsigned fixed point binary number with four bits before and six bits after the binary point.	
	Figure 1	
	0 1 1 1 0 1 1 0	
	Convert the bit pattern in Figure 1 to decimal.	
	[2 marks]	
		8



0 3.1	State the name of the component on a sound card that transforms the continuous signal received from a microphone to a form that can be stored by a computer. [1 mark]
0 3.2	A bitmap image is 52 pixels in height and 26 pixels in width. The bitmap representation of the image requires 845 bytes. Calculate the maximum number of colours that could be used in the bitmap image. You should show all your working. [2 marks]
0 3.3	When a bitmap image is stored in a file, additional information is stored as well as the colours of the pixels. For example, the bitmap file might contain information on the date of creation, image width and height. State the name given to this additional information when storing a bitmap image.
	[1 mark] Question 3 continues on the next page



0 3.4	A sound is recorded with a sample rate of 96 000 Hz and a sample resolution of 24 bits. The file size of the recording is 12 096 kilobytes.
	A sample rate of 1 Hz means that one sample has been taken every second.
	Calculate the duration of the sound recording.
	You should show all your working. [3 marks]
0 3.5	A sample resolution of 16 bits is commonly used in audio recordings.
	Explain why increasing the sample resolution from 16 bits to 24 bits can improve the quality of an audio recording.
	[1 mark]



		Do not write
0 3.6	MIDI does not use sampling to represent music.	outside the box
	Describe how music is represented using MIDI. [2 marks]	
0 3.7	Explain one advantage of using MIDI instead of sampled sound to represent music. [1 mark]	
		11
	Turn over for the next question	

0 4 . 1	A message is encrypted using a Caesar cipher that operates with a shift variety for example, the letter ${\tt A}$ in plaintext would be represented by ${\tt E}$ in ciphertext	
	The ciphertext for the message is WSSDI.	
	What is the plaintext for the message?	[1 mark]
0 4 . 2	Explain two reasons why Caesar ciphers are vulnerable to being cracked.	[2 marks]



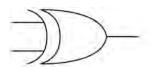
0 5.1			o indicate which of the following is an ex	ample of		Do not write outside the box
	system softwa	are.			[1 mark]	
		Α	Computer game	0		
		В	Image editor	0		
		С	Programming language translator	0		
		D	Video conferencing software	0		
		E	Word processor	0		
0 5.2	associated wi	th a con	manages hardware resources, for exam nputer system. uples of hardware resources that an oper			
	responsible fo	or mana	ging.		[2 marks]	
						3

Turn over for the next question



0 6. 1 Figure 2 shows the symbol for a logic gate.

Figure 2

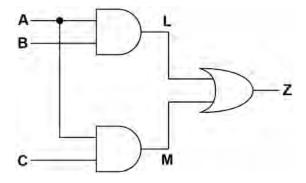


State the name of the logic gate shown in Figure 2.

[1 mark]

0 6.2 Figure 3 shows a logic circuit.

Figure 3



Complete the truth table for the logic circuit in Figure 3.

[2 marks]

Α	В	С	L	М	Z
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



Figure 4

Figure 4

A

C

Write a Boolean expression for Q. [3 marks]

Question 6 continues on the next page



			Do not write outside the		
0 6 . 4	Using the rules of Boolean algebra, simplify the following expression.				
	$\overline{W} \cdot X \cdot Z + W \cdot Z + X \cdot Y \cdot \overline{Z} + \overline{W} \cdot X \cdot Y \cdot 1$				
	$VV \cdot X \cdot Z + VV \cdot Z + X \cdot Y \cdot Z + VV \cdot X \cdot Y \cdot Y$				
	You must show your working.				
		[4 marks]			
	, 				
	Final anguar		10		
	Final answer				



0 7 . 1 Figure 5 shows some of the processor registers and buses that are used during the fetch stage of the fetch-execute cycle, together with the main memory. Figure 5 2 Processor Main memory 3 State the name of the components that are labelled in **Figure 5** with the numbers 1 to 4. In the case of register names, the full names **must** be stated. [2 marks] 1 2 3 4 Describe the stored program concept. [2 marks] Question 7 continues on the next page



0 7.3	In a particular processor instruction set, each instruction consists of an opcode and an operand. An operand could be an immediate value to be used by a program. State two other types of value that can be stored in an operand. [2 marks]	
0 7.4	Computer A and Computer B both have a processor with a clock speed of 2.8 GHz but Computer A performs tasks much faster than Computer B. Computer A has a larger cache and greater word length than Computer B.	
	Explain why the larger cache and greater word length are possible factors for the performance difference between Computer A and Computer B.	
	[2 marks]	
	Larger cache	
	Greater word length	
		8



Do not write outside the box Turn over for the next question DO NOT WRITE ON THIS PAGE ANSWER IN THE SPACES PROVIDED



Table 1 shows the standard AQA assembly language instruction set that should be used to answer question **0 8**

Table 1 - standard AQA assembly language instruction set

LOR Rd, <memory ref=""> Cad the value stored in the memory location specified by <memory ref=""> into register d. Store the value that is in register d into the memory location specified by <memory ref="">. ADD Rd, Rn, <operand2> Add the value specified in <operand2> to the value in register n and store the result in register d. SUB Rd, Rn, <operand2> Subtract the value specified by <operand2> from the value in register n and store the result in register d. MOV Rd, <operand2> Copy the value specified by <operand2> into register d. CMP Rn, <operand2> Copy the value stored in register n with the value specified by <operand2> into register d. CMP Rn, <operand2> Copy the value stored in register n with the value specified by <operand2>. B <label> Always branch to the instruction at position <label> in the program. B<condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are: EQ: equal to NE: not equal to GT: greater than LT: less than AND Rd, Rn, <operand2> Perform a bitwise logical AND operation between the value in register n and the value specified by <operand2> and store the result in register d. Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register n and the value specified by <operand2> and store the result in register n. EOR Rd, Rn, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Coperand2> into the program.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></condition></condition></label></label></condition></label></label></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></memory></memory></memory>		
specified by <memory ref="">. ADD Rd, Rn, <operand2> Add the value specified in <operand2> to the value in register n and store the result in register d. SUB Rd, Rn, <operand2> Copy the value specified by <operand2> into register d. MOV Rd, <operand2> Copy the value specified by <operand2> into register d. CMP Rn, <operand2> Compare the value stored in register n with the value specified by <operand2>. Always branch to the instruction at position <label> in the program. B<condition> <label> Branch to the instruction at position <label> in the program. B<condition> <label> Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the instruction at position <label> in the program. Branch to the</label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></label></condition></label></label></condition></label></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></memory>	LDR Rd, <memory ref=""></memory>	Load the value stored in the memory location specified by <memory ref=""> into register d.</memory>
Add the value specified in <pre>operand2> to the value in register n and store the result in register d. SUB Rd, Rn, <pre>coperand2> Subtract the value specified by <pre>coperand2> from the value in register n and store the result in register d. MOV Rd, <pre>coperand2> Copy the value specified by <pre>coperand2> into register d. CMP Rn, <pre>coperand2> Compare the value stored in register n with the value specified by <pre>coperand2> into register n with the value specified by <pre>coperand2> into register n with the value specified by <pre>coperand2> Always branch to the instruction at position <label> in the program. B</label></pre> B</pre> B</pre> B</pre> B</pre> B</pre> B</pre> B</pre> B</pre> B B Compare the value stored in register n with the value specified by <pre>comparison</pre> B Compare the value stored in register n with the value specified by <pre>comparison</pre> B B Condition> <label> in the program. B B B B B B B Condition> label Branch to the instruction at position label in the program. B B B B B Condition> label Branch to the instruction at position label in the program. B B B B Condition> label Branch to the instruction at position label in the program. B B B Condition> label Int label Int label Int label Branch to the instruction at position label Int label Int label Branch to the instruction at position label Int label Int label Branch to the instruction at position label Int label Branch to the instruction at position label Int label Branch to the instruction at position label Int label Branch to th</label>	STR Rd, <memory ref=""></memory>	Store the value that is in register d into the memory location
register n and store the result in register d. SUB Rd, Rn, <operand2> Subtract the value specified by <operand2> from the value in register n and store the result in register d. MOV Rd, <operand2> Copy the value specified by <operand2> into register d. CMP Rn, <operand2> Compare the value stored in register n with the value specified by <operand2>. Always branch to the instruction at position <label> in the program. B<condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are: EQ: equal to NE: not equal to GT: greater than LT: less than AND Rd, Rn, <operand2> Perform a bitwise logical AND operation between the value in register n and the value specified by <operand2> and store the result in register d. ORR Rd, Rn, <operand2> Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register n. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register n. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register n.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></condition></condition></label></label></condition></label></operand2></operand2></operand2></operand2></operand2></operand2>		specified by <memory ref="">.</memory>
SUB Rd, Rn, <operand2> Subtract the value specified by <operand2> from the value in register n and store the result in register d. MOV Rd, <operand2> Copy the value specified by <operand2> into register d. CMP Rn, <operand2> Copy the value stored in register n with the value specified by <operand2>. B <label> Always branch to the instruction at position <label> in the program. B<condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are:</condition></condition></label></label></condition></label></label></operand2></operand2></operand2></operand2></operand2></operand2>	ADD Rd, Rn, <pre>coperand2></pre>	Add the value specified in <pre><pre><pre><pre></pre></pre> Add the value specified in <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
SUB Rd, Rn, <operand2> Subtract the value specified by <operand2> from the value in register n and store the result in register d. MOV Rd, <operand2> Copy the value specified by <operand2> into register d. CMP Rn, <operand2> Copy the value stored in register n with the value specified by <operand2>. B <label> Always branch to the instruction at position <label> in the program. B<condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are:</condition></condition></label></label></condition></label></label></operand2></operand2></operand2></operand2></operand2></operand2>		register n and store the result in register d.
in register n and store the result in register d. MOV Rd, <operand2> Copy the value specified by <operand2> into register d. CMP Rn, <operand2> Compare the value stored in register n with the value specified by <operand2>. B <label> Always branch to the instruction at position <label> in the program. B<condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are:</condition></condition></label></label></condition></label></label></operand2></operand2></operand2></operand2>	SUB Rd, Rn, <pre>coperand2></pre>	· ·
CMP Rn, <operand2> Compare the value stored in register n with the value specified by <operand2>. B <label> Always branch to the instruction at position <label> in the program. B<condition> <label> B<condition> <label> B<condition> <label> B<condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are:</condition></condition></label></label></condition></label></condition></label></condition></label></condition></label></label></operand2></operand2>	_	· · · · · · · · · · · · · · · · · · ·
specified by <operand2>. B <label> Always branch to the instruction at position <label> in the program. B Always branch to the instruction at position <label> in the program. B<td>MOV Rd, <operand2></operand2></td><td>Copy the value specified by <operand2> into register d.</operand2></td></label></label></label></operand2>	MOV Rd, <operand2></operand2>	Copy the value specified by <operand2> into register d.</operand2>
specified by <operand2>. B <label> Always branch to the instruction at position <label> in the program. B Always branch to the instruction at position <label> in the program. B<td>CMP Rn, <operand2></operand2></td><td>Compare the value stored in register n with the value</td></label></label></label></operand2>	CMP Rn, <operand2></operand2>	Compare the value stored in register n with the value
the program. B <condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are:</condition></condition></label></label></condition>	_	· ·
the program. B <condition> <label> Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are:</condition></condition></label></label></condition>	B <label></label>	Always branch to the instruction at position <label> in</label>
Branch to the instruction at position <1abel> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are:</condition></condition>		
Possible values for <condition> and their meanings are:</condition>	B <condition> <label></label></condition>	1 0
EQ: equal to BE: not equal to GT: greater than LT: less than AND Rd, Rn, <operand2> Perform a bitwise logical AND operation between the value in register n and the value specified by <operand2> and store the result in register d. ORR Rd, Rn, <operand2> Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>		comparison met the criterion specified by <condition>.</condition>
AND Rd, Rn, <operand2> Perform a bitwise logical AND operation between the value in register n and the value specified by <operand2> and store the result in register d. ORR Rd, Rn, <operand2> Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>		Possible values for <condition> and their meanings are:</condition>
AND Rd, Rn, <operand2> Perform a bitwise logical AND operation between the value in register n and the value specified by <operand2> and store the result in register d. ORR Rd, Rn, <operand2> Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>		EQ: equal to NE: not equal to
register n and the value specified by <operand2> and store the result in register d. ORR Rd, Rn, <operand2> Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>		GT: greater than LT: less than
store the result in register d. ORR Rd, Rn, <operand2> Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>	AND Rd, Rn, <operand2></operand2>	Perform a bitwise logical AND operation between the value in
ORR Rd, Rn, <operand2> Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>		register n and the value specified by <pre><operand2></operand2></pre> and
register n and the value specified by <operand2> and store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>		store the result in register d.
store the result in register d. EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>	ORR Rd, Rn, <operand2></operand2>	· · · · · · · · · · · · · · · · · · ·
EOR Rd, Rn, <operand2> Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2></operand2>		· · · -
between the value in register n and the value specified by <operand2> and store the result in register d. MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2>		· · · · · · · · · · · · · · · · · · ·
<pre></pre>	EOR Rd, Rn, <operand2></operand2>	\ , , i
MVN Rd, <operand2> Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2></operand2></operand2>		The state of the s
specified by <operand2> and store the result in register d. LSL Rd, Rn, <operand2> Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d. LSR Rd, Rn, <operand2> Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.</operand2></operand2></operand2></operand2></operand2>		
LSL Rd, Rn, <pre>coperand2> Logically shift left the value stored in register n by the number of bits specified by <pre>coperand2> and store the result in register d.</pre> LSR Rd, Rn, <pre>coperand2> Logically shift right the value stored in register n by the number of bits specified by <pre>coperand2> and store the result in register d.</pre></pre></pre>	MVN Rd, <operand2></operand2>	
of bits specified by <pre>coperand2> and store the result in register d. LSR Rd, Rn, <pre>coperand2> Logically shift right the value stored in register n by the number of bits specified by <pre>coperand2> and store the result in register d.</pre></pre></pre>		
register d. LSR Rd, Rn, <pre>coperand2></pre> Logically shift right the value stored in register n by the number of bits specified by <pre>coperand2></pre> and store the result in register d.	LSL Rd, Rn, <operand2></operand2>	
LSR Rd, Rn, <pre>coperand2> Logically shift right the value stored in register n by the number of bits specified by <pre>coperand2> and store the result in register d.</pre></pre>		· · · -
number of bits specified by <operand2> and store the result in register d.</operand2>		
result in register d.	LSR Rd, Rn, <operand2></operand2>	
HALT Stops the execution of the program.		
	HALT	Stops the execution of the program.

Labels: A label is placed in the code by writing an identifier followed by a colon (:). To refer to a label the identifier of the label is placed after the branch instruction.

Interpretation of operand2>

<operand2> can be interpreted in two different ways, depending on whether the first character
is a # or an R:

- 1. # use the decimal value specified after the #, eg #25 means use the decimal value 25
- 2. Rm use the value stored in register m, eg R6 means use the value stored in register 6

The available general purpose registers that the programmer can use are numbered 0–12



		_
0 8	Registers R1 and R3 each store a different positive number.	Do not write outside the box
	Write a program using the standard AQA assembly language in Table 1 that will:	
	 store the greater of these two numbers in R1 store 1 in R2 if the value originally in R1 is greater than the value in R3, storing 3 in R2 otherwise. 	
	[4 marks]	
		-
		-
		.
		4

Turn over for the next question



0 9.1	RFID tags can be read by an RFID reader.	Do not write outside the box
	Describe how data is read from an RFID tag.	
	[3 marks]	
0 9.2	An RFID tag can be active or passive. A passive tag must be moved within a few centimetres of an RFID reader to be read. An active tag will have its own power source, which allows it to be read from a greater distance.	
	It is decided that RFID tags will be used in passports to store personal information.	
	Explain why passive tags are likely to be a more appropriate choice than active tags for use in passports.	
	[2 marks]	
		5



Do not write outside the box Turn over for the next question DO NOT WRITE ON THIS PAGE ANSWER IN THE SPACES PROVIDED



		Do not write
1 0	A smartphone company has designed a device that can be put into items such as luggage to help locate these items if they are lost. The device works by sending an encrypted signal containing its current location to nearby smartphones. The smartphones relay the signal to the company's servers via the Internet, allowing a user to see the device's exact location using a mobile phone app.	outside the box
	The company expects to sell hundreds of millions of devices. The data collected from each device will be permanently kept in secondary storage on the company's servers. The company is planning to use solid-state drives in the servers that will hold device location data but is unsure whether using solid-state drives is a good idea.	
	Discuss a range of moral, ethical, legal and cultural issues raised by the new device and explain the properties of solid-state drives that the company should consider when deciding on a secondary storage technology.	
	In your answer you will be assessed on your ability to follow a line of reasoning to	
	produce a coherent, relevant and structured response. [12 marks]	
	-	



Do not write outside the
box
_
_
_
_
_
_
_
_
_
12
_

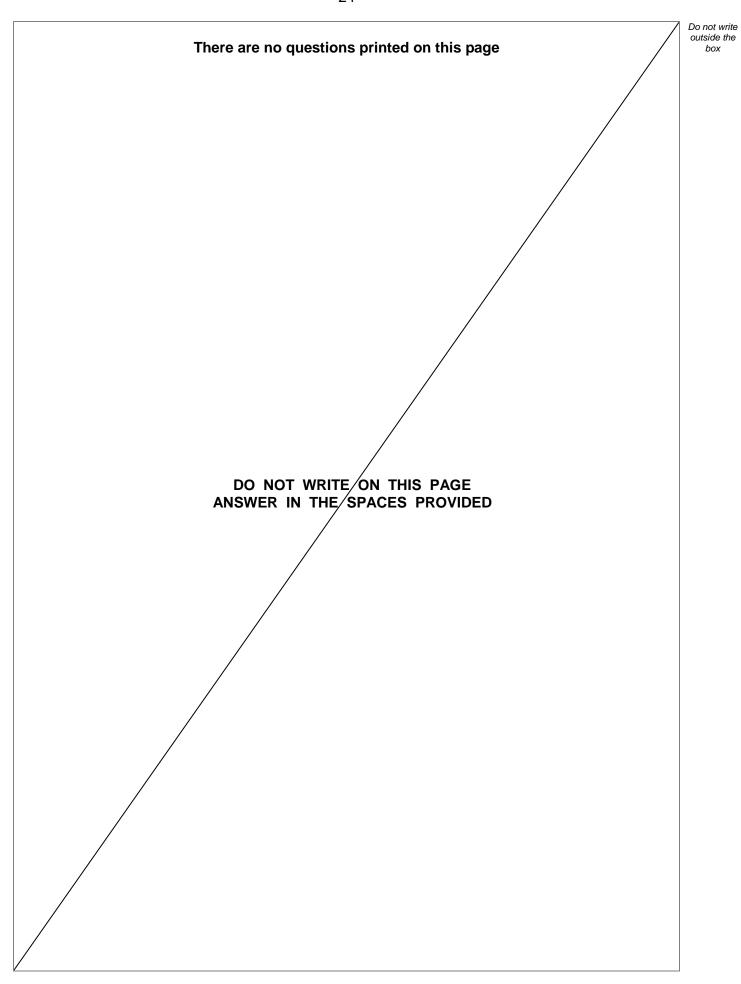


1.1	Describe the difference between baud rate and bit rate.	Do no outsi b
	[2 marks]	
1.2	A photographer wants to create a large file sharing network to allow thousands of photographers to share their photos with each other for free.	
	State two reasons why the photographer may choose to use a peer-to-peer network rather than a client-server network. [2 marks]	
1 . 3	Explain the purpose of a Service Set Identifier (SSID).	
	[1 mark]	



1 1.4	Explain how disabling SSID broadcasting can increase the security of a wireless network. [2 marks]	Do not write outside the box
		_
		_
		- - <u>-</u>
		_ 7
	END OF QUESTIONS	







Question number	Additional page, if required. Write the question numbers in the left-hand margin.

	<u></u>
	3



Question number	Additional page, if required. Write the question numbers in the left-hand margin.



Question number	Additional page, if required. Write the question numbers in the left-hand margin.

	<u></u>
	3



28 There are no questions printed on this page DO NOT WRITE ON THIS PAGE ANSWER IN THE SPACES PROVIDED

Copyright information

For confidentiality purposes, all acknowledgements of third-party copyright material are published in a separate booklet. This booklet is published after each live examination series and is available for free download from www.aqa.org.uk.

Permission to reproduce all copyright material has been applied for. In some cases, efforts to contact copyright-holders may have been unsuccessful and AQA will be happy to rectify any omissions of acknowledgements. If you have any queries please contact the Copyright Team.

Copyright © 2024 AQA and its licensors. All rights reserved.



