A

AQA

A-level
COMPUTER SCIENCE
Paper 2
7517/2

Insert

FIGURE 1 for use in answering Question 03.1

FIGURE 3 for use in answering Question 04.1

FIGURE 4 for use in answering Question 04.2

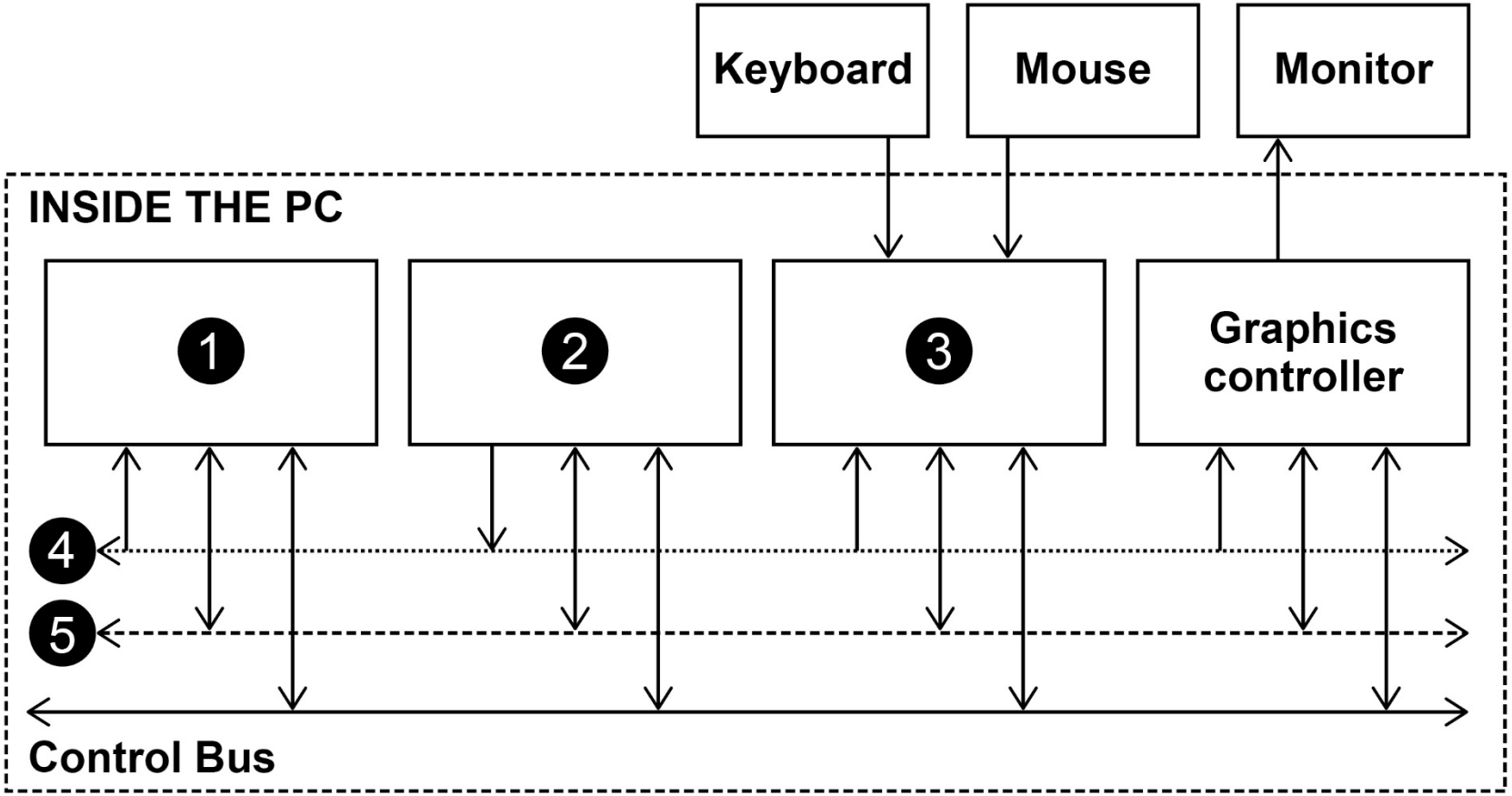
FIGURE 5 for use in answering Question 05

FIGURE 7 for use in answering Question 06

TABLE 2 for use in answering Question 09.3

[Turn over]

FIGURE 1


Keyboard
Mouse
Monitor
INSIDE THE PC
1
2
3
Graphics controller
4
5
Control Bus

BLANK PAGE

[Turn over]

FIGURE 3

Property(PropertyID, HouseNum, Street, Area, Postcode, Bedrooms, Bathrooms, AskingPrice, SellerID)

Seller(SellerID, Title, Forename, Surname, Telephone)

Buyer(BuyerID, Title, Forename, Surname, Telephone, DesiredArea, MinBedrooms, MaxPrice)

Viewing(BuyerID, PropertyID, ViewingDate, ViewingTime)

Sale(SaleID, PropertyID, BuyerID, SalePrice)

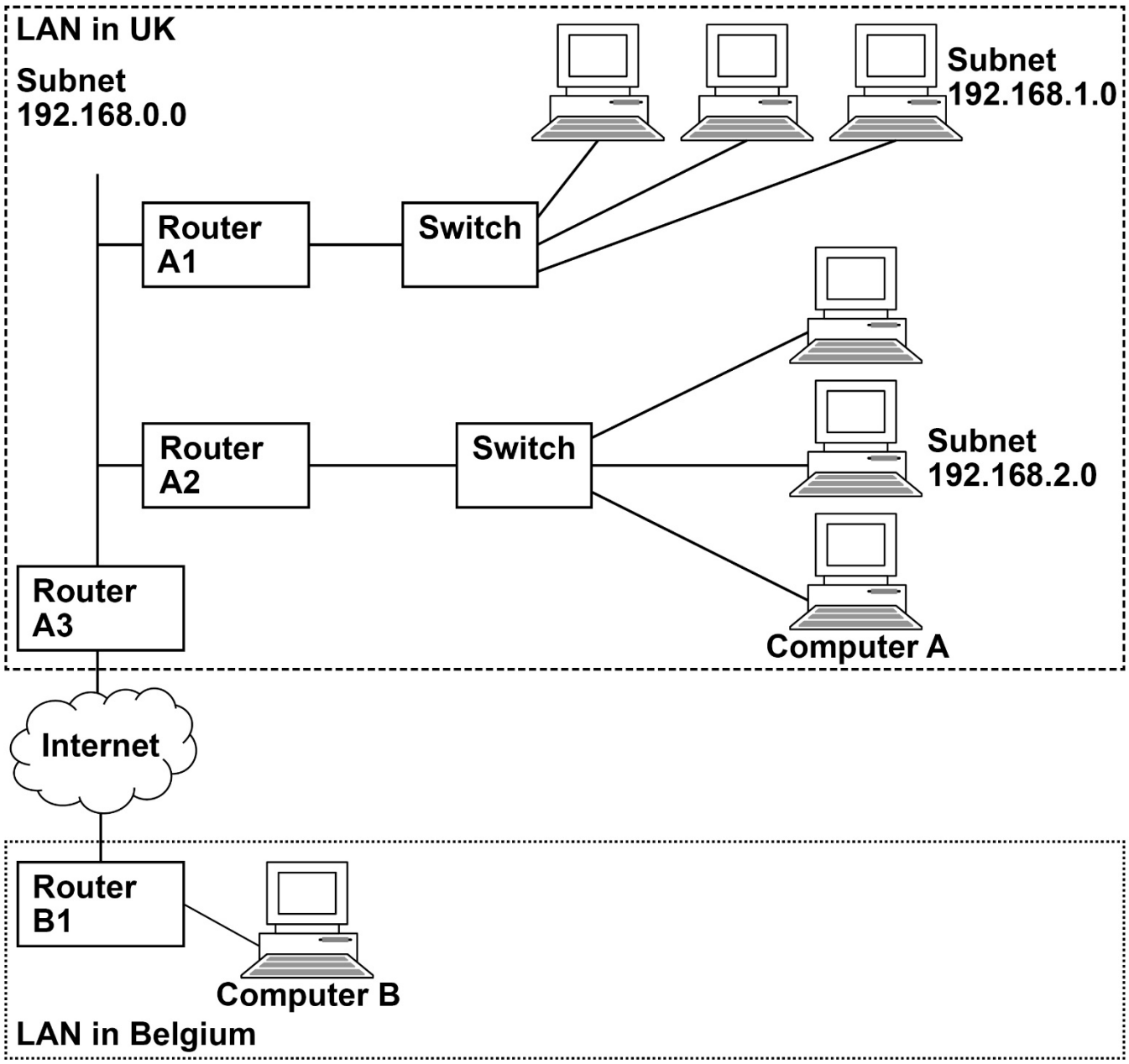
- **The Property relation stores details of the properties that are for sale. This includes the number of bedrooms and the number of bathrooms that a property has.**
- **The Seller relation stores details of people who are selling the properties.**
- **The Buyer relation stores details of the people who are looking to buy a property and information about the type of property they want, including the area that they want to live in, the minimum number of bedrooms that they need in a property and the maximum price that they are prepared to pay.**
- **An entry is made in the Viewing relation whenever a buyer arranges to look at a property.**
- **An entry is made in the Sale relation whenever a property is sold to a buyer. The SalePrice may be different to the AskingPrice for the property.**

[Turn over]

FIGURE 4

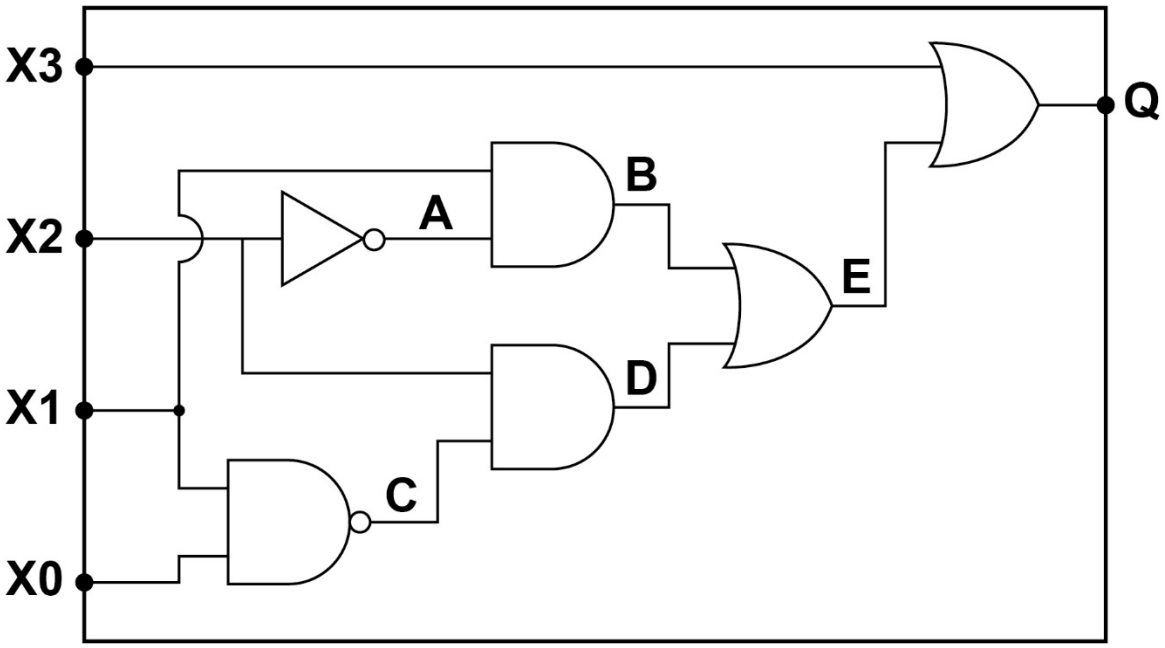
Representation 1	Representation 2
{"Properties":[{ "PropertyID": 8026, "HouseNum": "12", "Street": "Chester Drive", "Bedrooms": 4 }, { "PropertyID": 9034, "HouseNum": "23a", "Street": "Castle Street", "Bedrooms": 5 }]}	<Properties> <Property> <PropertyID>8026</PropertyID> <HouseNum>12</HouseNum> <Street>Chester Drive</Street> <Bedrooms>4</Bedrooms> </Property> <Property> <PropertyID>9034</PropertyID> <HouseNum>23a</HouseNum> <Street>Castle Street</Street> <Bedrooms>5</Bedrooms> </Property> </Properties>

FIGURE 5



[Turn over]

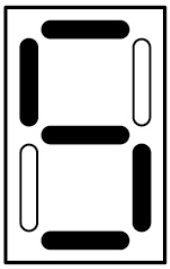
FIGURE 7



The inputs to the circuit are X3, X2, X1 and X0. Together these form the binary representation of the decimal digit to display. For example, if the inputs to the circuit were:

X3	X2	X1	X0
0	1	0	1

then the display would need to show this pattern:



as 0101 is the binary representation of the decimal digit 5

The output Q is connected to one segment of the display. When Q is 1 this segment lights up, when it is 0 the segment does not light up.

[Turn over]

TABLE 2
Standard AQA assembly language instruction set

`LDR Rd, <memory ref>`	**Load the value stored in the memory location specified by** `<memory ref>` **into register** `d`**.**
`STR Rd, <memory ref>`	**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>`	**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:** `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** 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**.**
HALT	**Stops the execution of the program.**

END OF SOURCE MATERIAL

BLANK PAGE