A-level

COMPUTER SCIENCE

Paper 2

7517/2

Insert

[Turn over]

FIGURE 1

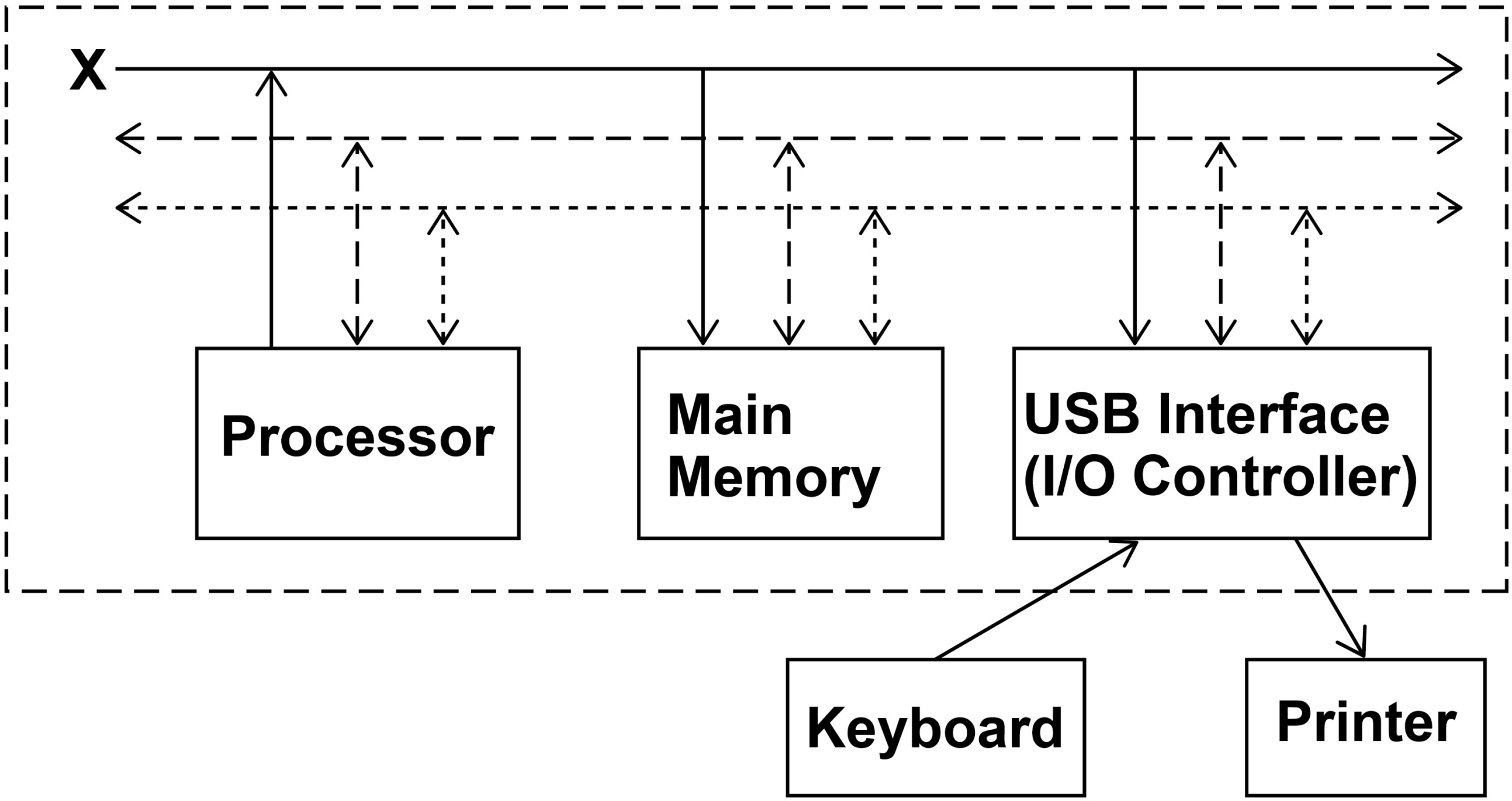

X
Processor
Main Memory
USB Interface (I/O Controller)
Keyboard
Printer

FIGURE 2

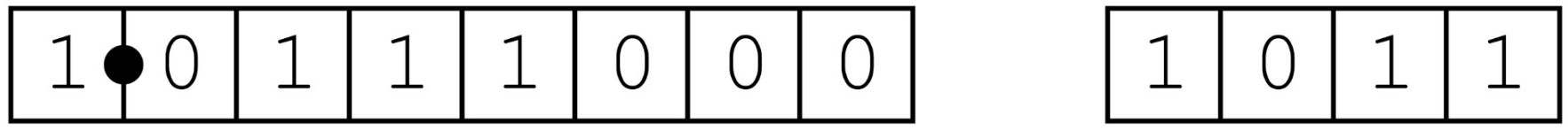
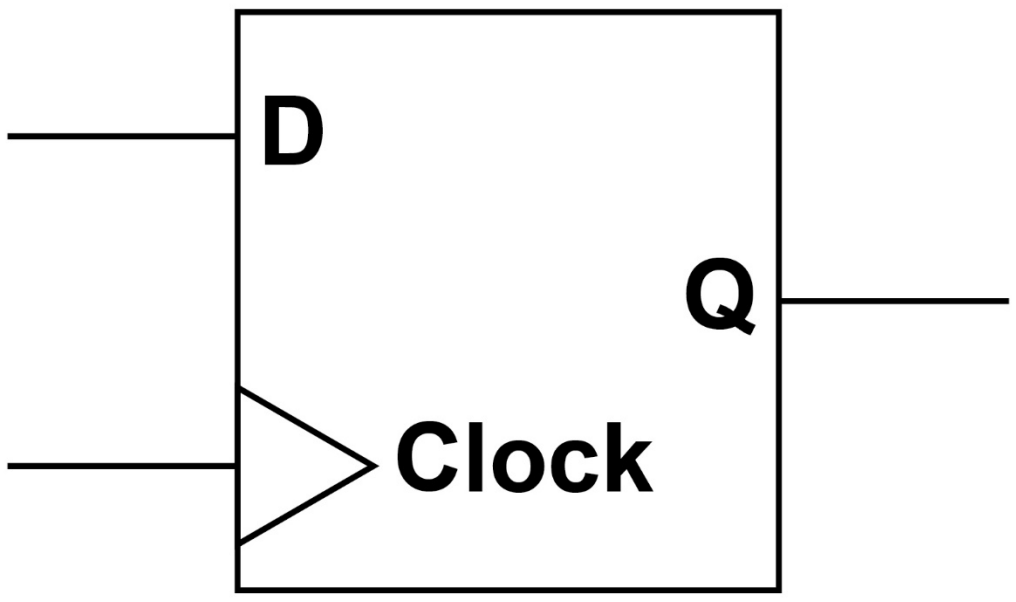


FIGURE 3



[Turn over]

FIGURE 5

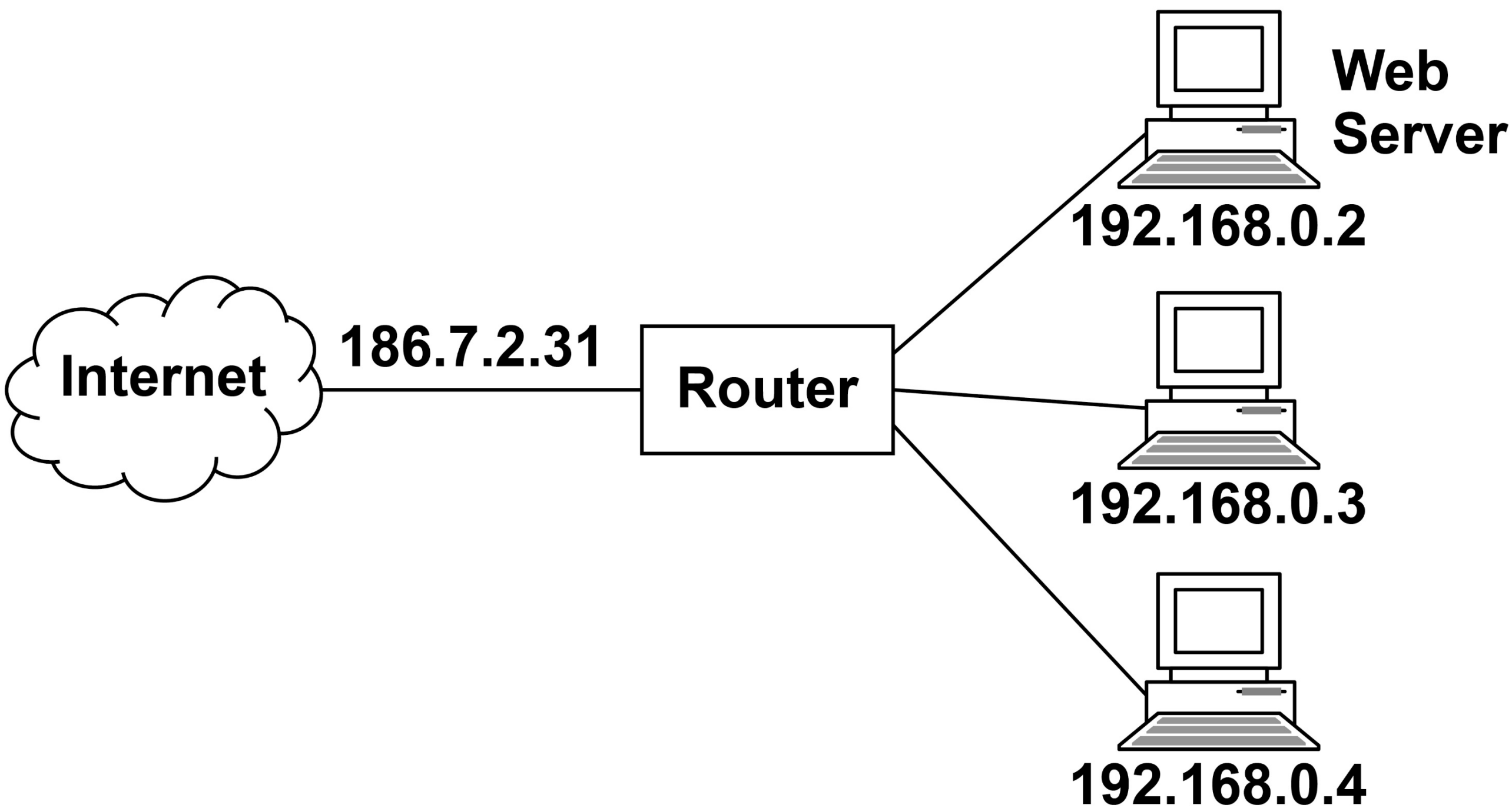

Web
Server
192.168.0.2
Internet
186.7.2.31
Router
192.168.0.3
192.168.0.4

FIGURE 6

Product(<u>ProductID</u>, Description, QuantityInStock, SupplierID)

Sale(<u>SaleID</u>, CustomerID, SaleDate)

SaleLine(<u>SaleID, ProductID</u>, QuantitySold)

Customer(<u>CustomerID</u>, Forename, Surname, EmailAddress)

Supplier(<u>SupplierID</u>, SupplierName, SupplierEmail)

[Turn over]

TABLE 1

This table is included so that you can answer Question 10.1

TABLE 1 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`

[Turn over]

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.**

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:**

- `#` **– use the decimal value specified after the** `#`**, eg** `#25` **means use the decimal value 25**
- `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

[Turn over]

FIGURE 7

```
   LDR R1, 130
   MOV R2, #0
   MOV R4, #0
repeat:
   ADD R2, R2, #1
   AND R3, R1, #1
   CMP R3, #0
   BEQ skip
   ADD R4, R4, #1
skip:
   LSR R1, R1, #1
   CMP R2, #7
   BNE repeat
   LDR R1, 130
   AND R4, R4, #1
   CMP R4, #0
   BNE else
```

```
   ORR R1, R1, #128
   B end
else:
   AND R1, R1, #127
end:
   STR R1, 130
   HALT
```

The program performs a task on a value stored in memory location 130. The value in this memory location is a 7-bit ASCII code.

For example, if memory location 130 was used to store the ASCII character ‘S’ then it would contain the value 83, which in binary is:

0	1	0	1	0	0	1	1

END OF INSERT

BLANK PAGE