4 The table shows part of the instruction set for a processor. The processor has one register, the Accumulator (ACC).

Instruction		Explanation
Opcode	**Operand**	
AND	#n / Bn / &n	Bitwise AND operation of the contents of the ACC with the operand
AND	<address>	Bitwise AND operation of the contents of the ACC with the contents of <address>
XOR	#n / Bn / &n	Bitwise XOR operation of the contents of the ACC with the operand
XOR	<address>	Bitwise XOR operation of the contents of the ACC with the contents of <address>
OR	#n / Bn / &n	Bitwise OR operation of the contents of the ACC with the operand
OR	<address>	Bitwise OR operation of the contents of the ACC with the contents of <address>
LSL	#n	Bits in ACC are shifted logically n places to the left. Zeros are introduced on the right-hand end.
LSR	#n	Bits in ACC are shifted logically n places to the right. Zeros are introduced on the left-hand end.
<address> can be an absolute or symbolic address # denotes a denary number, e.g. #127 B denotes a binary number, e.g. B10010001 & denotes a hexadecimal number, e.g. &4A		

(a) The ACC currently contains the following positive binary integer:

0	0	0	1	1	1	1	0

Write a bit manipulation instruction that uses a binary shift to change the contents of the ACC to:

0	1	1	1	1	0	0	0

Instruction ..... [1]

(b) The ACC currently contains the following positive binary integer:

1	1	1	0	0	0	1	1

Write the contents of the ACC after the instruction `XOR &12` is carried out.

[1]

(c) The ACC currently contains the following positive binary integer:

1	1	1	0	0	0	1	1

Write the contents of the ACC after the instruction `AND #63` is carried out.

[1]

(d) The ACC currently contains the following positive binary integer:

1	1	1	0	0	0	1	1

The current contents of memory are:

Address	**Data**
98	00100100
99	00110001
100	00110011
101	10100011
102	10101100

Write the contents of the ACC after the instruction `OR 100` is carried out.

[1]