**5** The following table shows part of the instruction set for a processor. The processor has two registers: the Accumulator (ACC) and an Index Register (IX).

| Instruction | | Explanation |
|---|---|---|
| **Opcode** | **Operand** | |
| LDM | #n | Immediate addressing. Load the number n to ACC |
| LDD | <address> | Direct addressing. Load the contents of the location at the given address to ACC |
| LDI | <address> | Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC |
| LDX | <address> | Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC |
| LDR | #n | Immediate addressing. Load the number n to IX |
| ADD | #n/Bn/&n | Add the number n to the ACC |
| ADD | <address> | Add the contents of the given address to the ACC |
| SUB | #n/Bn/&n | Subtract the number n from the ACC |
| SUB | <address> | Subtract the contents of the given address from the ACC |
| INC | <register> | Add 1 to the contents of the register (ACC or IX) |
| <address> can be an absolute or a symbolic address<br># denotes a denary number, e.g. #123<br>B denotes a binary number, e.g. B01001010<br>& denotes a hexadecimal number, e.g. &4A | | |

**(a)** The current contents of memory are shown:

| Address | Data |
|---|---|
| **10** | 1 |
| **11** | 3 |
| **12** | 5 |
| **13** | 11 |
| **14** | 10 |
| **15** | 16 |
| **16** | 12 |

The current contents of the ACC and IX are shown:

| | |
|---|---|
| **ACC** | 10 |
| **IX** | 0 |

Complete the table by writing the content of the ACC after each program has run.

| Program number | Code | ACC content |
|---|---|---|
| 1 | LDI 15<br>SUB #1 | |
| 2 | LDD 14<br>ADD 11 | |
| 3 | LDM #11<br>ADD #3<br>SUB 16 | |
| 4 | LDR #2<br>LDX 14<br>ADD #2 | |

[4]

**(b)** The processor includes these bit manipulation instructions:

| Instruction | | Explanation |
|---|---|---|
| **Opcode** | **Operand** | |
| AND | #n/Bn/&n | Bitwise AND operation of the contents of ACC with the operand |
| AND | <address> | Bitwise AND operation of the contents of ACC with the contents of <address> |
| XOR | #n/Bn/&n | Bitwise XOR operation of the contents of ACC with the operand |
| XOR | <address> | Bitwise XOR operation of the contents of ACC with the contents of <address> |
| OR | #n/Bn/&n | Bitwise OR operation of the contents of ACC with the operand |
| OR | <address> | Bitwise OR operation of the contents of ACC with the contents of <address> |
| <address> can be an absolute or a symbolic address<br># denotes a denary number, e.g. #123<br>B denotes a binary number, e.g. B01001010<br>& denotes a hexadecimal number, e.g. &4A | | |

The current contents of memory are shown:

| Address | Data |
|---|---|
| **25** | 11000110 |
| **26** | 11100001 |
| **27** | 10000001 |
| **28** | 11001101 |
| **29** | 00001111 |

The current content of the ACC is shown:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |

Complete the table by writing the content of the ACC after each program has run.

The binary number 01000110 is reloaded into the ACC before each program is run.

| Program number | Code | ACC content |
|---|---|---|
| 1 | XOR 29 | |
| 2 | AND #29 | |
| 3 | OR B11111111 | |

[3]