8 **(a)** 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
INC	<register>	Add 1 to the contents of the register (ACC or IX)
STO	<address>	Store the contents of ACC at the given address
ADD	#n/Bn/&n	Add the number n to the ACC
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX)
JMP	<address>	Jump to the given address
CMP	<address>	Compare the contents of ACC with the contents of <address>
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
END		Return control to the operating system

ACC denotes Accumulator
<address> can be an absolute or a symbolic address
# denotes a denary number, e.g. #123
B denotes a binary number, e.g. B01001010
& denotes a hexadecimal number, e.g. &4A

The current contents of memory are:

address	Instruction
80	10
81	8
82	80
83	81
...	
200	LDD 81
201	INC ACC
202	STO 83
203	LDI 82
204	CMP 83
205	JPE 209
206	LDD 83
207	ADD #10
208	JMP 210
209	DEC ACC
210	STO 81
211	END

Trace the program currently in memory using the following trace table.

Instruction address	ACC	Memory address			
		80	**81**	**82**	**83**
		10	8	80	81

[4]

(b) 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 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>
`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. #123 B denotes a binary number, e.g. B01001010 & denotes a hexadecimal number, e.g. &4A		

(i) Write the bit manipulation instruction that can be used to set the least significant bit to 1 in an 8-bit register. All other bits must remain unchanged.

The instruction needs to work on a register that contains any 8-bit binary number.

..... [1]

(ii) The ACC currently contains the following binary value.

0	1	0	1	0	1	0	1

Write the result after the instruction `XOR &FE` is run.

[1]

(iii) The ACC currently contains the following binary value.

0	1	1	0	1	0	1	1

Write the result after the instruction `LSR #5` is run.

[1]

7 **(a)** A computer can perform logical and arithmetic shifts.

(i) Show the result of a logical left shift of 2 places on the two's complement binary integer 11001010

.....

..... [1]

(ii) Show the result of an arithmetic right shift of 3 places on the two's complement binary integer 10011110

.....

..... [1]

(b) Complete the following binary addition. Show your working. Include any overflow bit(s).

```
  1 1 1 1 0 1 0 1
+ 1 0 1 1 0 0 0 1
-----------------
```

[2]

(c) Subtract the binary number 00011110 from the binary number 01100100 using binary subtraction. Show your working.

[1]

7 The following table shows part of the instruction set for a processor. The processor has two registers, the Accumulator (ACC) and the 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)
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX)

<address> can be an absolute or a symbolic address
# denotes a denary number, e.g. #123
B denotes a binary number, e.g. B01001010
& denotes a hexadecimal number, e.g. &4A

(a) The current contents of memory are shown:

Address	Data
50	54
51	55
52	50
53	52
54	100
55	25
56	50

The current contents of the ACC and IX are shown:

ACC	50
IX	45

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

	Instructions	ACC content

1	`LDD 50`
	`ADD #4`
	`ADD 54`	

2	`LDI 53`
	`DEC ACC`
	`ADD 56`	

3	`LDM #55` `SUB #5`	

[3]

(b) The instruction set also 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
# denotes a denary number, e.g. #123
B denotes a binary number, e.g. B01001010
& denotes a hexadecimal number, e.g. &4A

Explain how bit manipulation can be used to clear the data in an 8-bit register.

Write the bit manipulation instruction that will be used.

Explanation .....

.....

.....

.....

.....

Instruction .....

[3]

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
# denotes a denary number, e.g. #123
B denotes a binary number, e.g. B01001010
& 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 SUB #1	
2	LDD 14 ADD 11	
3	LDM #11 ADD #3 SUB 16	
4	LDR #2 LDX 14 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 # denotes a denary number, e.g. #123 B denotes a binary number, e.g. B01001010 & 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]