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]

3 (a) (i) State what is meant by relative addressing.

.....

.....

..... [1]

(ii) Registers such as the Accumulator (ACC) and the Index Register (IX) are used in the CPU.

Identify **two** special purpose registers used in the CPU. Do **not** include the ACC or IX in your answers.

1 .....

2 .....

[2]

(b) The following table shows part of the instruction set for a processor. The processor has two registers: the ACC and an 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
<address> can be an absolute or symbolic address # denotes a denary number, e.g. #127		

The current contents of the main memory and the index register are shown.

Address	Instruction
98	8
99	16
100	3
101	98
102	32
IX	2

Write the contents of the ACC after each instruction is executed.

Instruction	Value in ACC
LDM #98	
LDI 101	
LDX 100	

[3]

(c) A student buys a new computer. The table shows the specifications of the old computer and the new computer.

Old computer	New computer
1.8 GHz dual core processor	2.3 GHz dual core processor
16 MB cache	32 MB cache

Explain why increasing the clock speed **and** increasing the cache memory will improve the performance of the computer.

Clock speed .....

.....

.....

.....

Cache memory .....

.....

.....

.....

[4]

6 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 the ACC
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>
CMP	#n	Compare the contents of the ACC with number n
CMP	<address>	Compare the contents of the ACC with the contents of <address>
LSL	#n	Bits in the ACC are shifted logically n places to the left. Zeros are introduced on the right-hand end
LSR	#n	Bits in the 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		

The current contents of main memory are shown:

Address	Data
100	0000 0011
101	1010 1110
102	1100 1100
103	1111 1111
104	1100 1100

(a) Complete the table by writing the contents of the ACC after the execution of each instruction.

Current contents of the ACC	Instruction	Contents of the ACC after the execution of the instruction
0000 1111	AND 101	
0000 0000	LDM #100	
0000 0001	XOR &F1	
0001 0001	CMP 101	

[4]

(b) The Von Neumann model for a computer system uses registers.

Describe the role of the Memory Address Register (MAR) and Memory Data Register (MDR) in the fetch-execute (F-E) cycle.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) Assembly language instructions are grouped.

Complete each statement by writing the name of the appropriate instruction group.

Loading data into the accumulator is an example of an instruction in the

..... group. Incrementing the index register is an example of an instruction in the ..... group. Branching to another address is an example of an instruction in the ..... group.

[3]

11 The table shows assembly language instructions for a processor that has one register, the Accumulator (ACC).

Label	Instruction		Explanation
	Opcode	**Operand**	
	LDM	#n	Load the number n to the ACC
	LDD	<address>	Load the contents of the location at the given address to ACC
	LDI	<address>	The address to be used is at the given address. Load the contents of this second address to the ACC.
	ADD	<address>	Add the contents of the given address to the ACC
	SUB	<address>	Subtract the contents of the given address from the ACC
	STO	<address>	Store the contents of the ACC at the given address
<label>:		<data>	Gives a symbolic address <label> to the memory location with the contents <data>
# denotes a denary number, e.g. #123 <label> can be used in place of <address>			

The current contents of memory are:

Address	Contents
150	26
300	86
420	150

Write **assembly language** code, using **only** the given instruction set to:

- store the contents of location 300 as labelled variable `A`
- store the contents of location 420 as labelled variable `B`
- add the value stored in the address contained in variable `B` to the value contained in variable `A`
- store the result in variable `Answer`.

Show the initialisation and values of the variables `A`, `B` and `Answer` in the table provided.

.....

.....

.....

.....

.....

.....

.....

.....

Label	Content

[7]

7 **(a)** The following table shows part of the instruction set for a processor. The processor has one register: the Accumulator (ACC).

Instruction		Explanation
Opcode	**Operand**	
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC
LDM	#n	Immediate addressing. Load the number n to ACC
STO	<address>	Store the contents of ACC at the given address
ADD	#n/Bn/&n	Add the number n to the ACC
ADD	<address>	Add the contents of the given address to the ACC
INC	<register>	Add 1 to the contents of the register (ACC)
CMP	<address>	Compare the contents of ACC with the contents of <address>
CMP	#n	Compare the contents of ACC with number n
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
JMP	<address>	Jump to the given address
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
10	12
11	11
12	10
13	22
14	22
…	
100	LDD 10
101	ADD 12
102	STO 11
103	CMP 14
104	JPE 107
105	INC ACC
106	JMP 101
107	STO 10
108	END

(i) Trace the program currently in memory using the following trace table.

Instruction address	ACC	Memory address				
		10	**11**	**12**	**13**	**14**
		12	11	10	22	22

[3]

(ii) State the effect of changing instruction `LDD 10` in address 100 to `LDM #10`

.....

.....

..... [1]

(iii) Identify **and** describe **one** mode of addressing **not** given in the table of instructions in part **(a)**.

Mode of addressing .....

Description .....

.....

.....

[2]

(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>
<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) The ACC currently contains the following binary value.

1	1	1	1	0	0	0	0

Write the result after the instruction `OR B00001111` is run.

[1]

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

0	0	0	1	1	1	0	1

Write the result after the instruction `XOR #30` is run.

[1]

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
STO	<address>	Store the contents of ACC at the given address
ADD	#n/Bn/&n	Add the number n to the ACC
ADD	<address>	Add the contents of the given address to the ACC
SUB	<address>	Subtract the contents of the given address from the ACC
SUB	#n/Bn/&n	Subtract the number n from 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>
CMP	#n	Compare the contents of ACC with number n
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

(a) The current contents of memory are:

Address	Instruction
50	47
51	48
52	49
53	50
...	
500	LDM #50
501	DEC ACC
502	CMP 51
503	JPE 505
504	JMP 501
505	STO 50
506	SUB #10
507	STO 51
508	END

(i) Trace the program currently in memory using the following trace table.

Instruction address	ACC	Memory address			
		50	**51**	**52**	**53**
		47	48	49	50

[3]

(ii) Complete the table by identifying **and** describing **two** modes of addressing that are **not** used in the program in part **(a)**.

Mode of addressing	Description
....	....
....	....

[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>
<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		

Write the bit manipulation instructions that can be used to set only the most significant bit to 1 in an 8-bit register.

The instructions need to work on a register that contains any 8-bit binary number.

.....

..... [2]

8 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
19	25
20	23
21	2
22	4
23	15
24	50
25	22

The current contents of the ACC and IX are shown:

ACC	50
IX	20

Complete the table by writing the content of the ACC and the IX after each set of instructions has run.

	Instructions	ACC content	IX content
1	`LDM #19` `DEC ACC`		
2	`LDD 23` `ADD 19`		
3	`LDI 25` `INC ACC`		
4	`LDR #21` `LDX 2`		

[5]

(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>
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 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 content of the ACC is shown:

ACC	1	0	0	1	1	0	1	0

(i) The table has three sets of instructions. The binary number 10011010 is reloaded into the ACC before each set of instructions is run.

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

	Instructions	**ACC content**
1	`LSL #2`	
2	`ADD #5` `AND #30`	
3	`OR B11110010` `INC ACC`	

[3]

(ii) Explain how bit manipulation can be used to test whether the binary number stored in the ACC represents an odd denary number.

Write the bit manipulation instruction that will be used.

Explanation .....

.....

.....

.....

.....

Instruction .....

[3]

4 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
19	24
20	2
21	1
22	3
23	5
24	4
25	22

The current contents of the ACC and IX are shown:

ACC	12
IX	1

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

Program number	Code	ACC content
1	`LDD 20` `ADD #2`	
2	`LDX 22`	

3	`LDI 25`
	`INC ACC`
	`SUB 22`	

4	`LDD 19`
	`LDM #5`
	`LDM #25`	

[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
30	01110101
31	11111111
32	00000000
33	11001100
34	10101010

The current content of the ACC is shown:

1	0	0	1	1	0	1	0

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

The binary number 10011010 is reloaded into the ACC before each program is run.

Program number	Code	ACC content
1	AND 31	
2	XOR B01001111	
3	OR #30	

[3]