4(a)	**1 mark** for: LSL #2
4(b)	1 1 1 1 0 0 0 1
4(c)	0 0 1 0 0 0 1 1
4(d)	1 1 1 1 0 0 1 1

3(a)(i)	**1 mark** • The value of the operand is an offset value which is added to another base value to give the address from which the contents are loaded to the accumulator
3(a)(ii)	**1 mark** per bullet point, **max 2** marks • Program counter (PC) • Memory Data Register (MDR) • Memory Address Register (MAR) • Current Instruction Register (CIR) • Status Register
3(b)	**1 mark** for each correct value, **max 3** marks

Instruction	Value in ACC
`LDM #98`	98
`LDI 101`	8
`LDX 100`	32

3(c)	**1 mark** per bullet point, **max 2** marks for each Clock Speed: • Processor can perform more F-E cycles per second • ... so more instructions / data can be processed each second Cache Memory: • Can store more of the most frequently used instructions • … which reduces the need to access slower RAM

<table>
<tr><td>6(a)</td><td>1 mark for each correct row

<table>
<tr><th>Current contents of the ACC</th><th>Instruction</th><th>Contents of the ACC after the execution of the instruction</th></tr>
<tr><td>0000 1111</td><td>AND 101</td><td>0000 1110</td></tr>
<tr><td>0000 0000</td><td>LDM #100</td><td>0110 0100</td></tr>
<tr><td>0000 0001</td><td>XOR &F1</td><td>1111 0000</td></tr>
<tr><td>0001 0001</td><td>CMP 101</td><td>0001 0001</td></tr>
</table>
</td></tr>
<tr><td>6(b)</td><td>1 mark for each bullet point, max 4 marks

• MAR stores the address of the next instruction / data to be read from / or written to memory
• The address is received from the Program Counter (PC)

• The MDR stores the data / instruction in the address stored in the MAR which has been read / written
• The instruction passes to the CIR for decoding and executing</td></tr>
<tr><td>6(c)</td><td>1 mark for each bullet point, max 3 marks

• Data movement
• Arithmetic operations
• Conditional and unconditional (jump) instructions</td></tr>
</table>

11

One mark per mark point (**Max 7**)
MP1 `LDD 300` seen
MP2 Correct use of `STO` seen (at least once)
MP3 Correct use of `LDD 420` seen
MP4 Correct use of `LDI B`
MP5 Correct use of `ADD A`

Opcode	Operand
LDD	300
STO	A
LDD	420
STO	B
LDI	B
ADD	A
STO	Answer

MP6 Correct labelling of three addresses `A:`, `B:` and `Answer:`
MP7 Correct contents in `A:` and `B:`
MP8 Correct value in `Answer:`

Label	Contents
A:	86
B:	150
Answer:	112

<table>
<tr><td>7(a)(i)</td><td>1 mark for each shaded section

<table>
<tr><td rowspan="2">Instruction address</td><td rowspan="2">ACC</td><td colspan="5">Memory address</td></tr>
<tr><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td></tr>
<tr><td></td><td></td><td>12</td><td>11</td><td>10</td><td>22</td><td>22</td></tr>
<tr><td>100</td><td>12</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>101</td><td>22</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>102</td><td></td><td></td><td>22</td><td></td><td></td><td></td></tr>
<tr><td>103</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>104</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>107</td><td></td><td>22</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>7(a)(ii)</td><td>1 mark from:

• The number 10 will be loaded into the ACC instead of 12 / contents of address 10
• The addition will give 20 not 22 so the comparison will fail and result in an infinite loop</td></tr>
<tr><td>7(a)(iii)</td><td>1 mark for name of mode, 1 mark for matching description

e.g.
• Indirect
• The operand points to the memory location that contains the address of the data

• Indexed
• The address of the data is formed by adding the contents of the Index Register (IX) to the operand

• Relative
• The address is calculated using its distance from a base address</td></tr>
<tr><td>7(b)(i)</td><td>1 mark for:

1111 1111</td></tr>
<tr><td>7(b)(ii)</td><td>1 mark for:

0000 0011</td></tr>
</table>

5(a)(i)

1 mark for each shaded section

Instruction address	ACC	Memory address			
		50	51	52	53
		47	48	49	50
500	50				
501	49				
502					
504					
501	48				
502					
503					
505		48			
506	38				
507			38		

5(a)(ii)

1 mark for addressing mode, **1 mark** for matching description

- Direct
- The operand is the address of the data

- Indirect
- The operand points to the memory location which is the address of the data

- Indexed
- The address of the data is formed by adding the contents of the Index Register (IX) to the operand

5(b)

1 mark for each correct instruction

e.g.
AND B00000000 / #0 / &0
OR B10000000 / #128 / &80

8(a)

1 mark for each correct content of the ACC (4)
1 mark for correct IX column

	Instructions	ACC content	IX content
1	`LDM #19` `DEC ACC`	**18**	**20**
2	`LDD 23` `ADD 19`	**40**	**20**
3	`LDI 25` `INC ACC`	**5**	**20**
4	`LDR #21` `LDX 2`	**15**	**21**

8(b)(i)

1 mark each correct content of ACC

	Instructions	ACC content
1	`LSL #2`	**0110 1000**
2	`ADD #5` `AND #30`	**0001 1110**
3	`OR B11110010` `INC ACC`	**1111 1011**

8(b)(ii)

1 mark each to **max 2** for explanation

- An odd binary number will have a 1 in the Least Significant Bit (LSB)
- A bit manipulation operation is required to access/mask only the LSB and clear all the others
- Compare the result of the masking with denary 1
- … the result of the comparison will be true if the number is odd

1 mark for correct instruction

- AND B00000001 // AND #1 // AND &01

4(a)

1 mark for each correct answer:

Program Number	Code	ACC Content
1	`LDD 20` `ADD #2`	**4**
2	`LDX 22`	**5**

3	`LDI 25`
	`INC ACC`
	`SUB 22`
		1

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

4(b)

1 mark for each correct answer:

Program Number	Code	ACC Content
1	`AND 31`	**1001 1010**
2	`XOR B01001111`	**1101 0101**
3	`OR #30`	**1001 1110**