Chapter 4
19-20
Solution

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>1 mark for two correct ticks, 2 marks for three correct ticks

<table>
<tr><th>Task</th><th>First pass</th><th>Second pass</th></tr>
<tr><td>Creation of symbol table</td><td>✓</td><td></td></tr>
<tr><td>Expansion of macros</td><td>✓</td><td></td></tr>
<tr><td>Generation of object code</td><td></td><td>✓</td></tr>
<tr><td>Removal of comments</td><td>✓</td><td></td></tr>
</table></td><td>2</td></tr>
<tr><td>5(b)</td><td>1 mark per bullet point to max 2

• Data movement
• Input and output of data
• Arithmetic operations
• Jump instructions
• Compare instructions</td><td>2</td></tr>
<tr><td>5(c)</td><td>1 mark per bullet point

• Storing 0 in 300 (line 21)
• Loading 65 (line 28)
• Outputting A (line 29)
• Loading 0 (line 30), incrementing ACC (line 31) and storing in 300 (line 32)
• Incrementing IX (line 33)
• Loading 67 (line 24) and adding 33 (line (25)
• Outputting d (line 26)
• Loading 1 (line 30), incrementing ACC (line 31), storing in 300 (line 32) and incrementing IX (line 33)</td><td>8</td></tr>
</table>

Instruction address	ACC	Memory address							IX	OUTPUT
		100	101	102	103	104	300	301		
		65	67	69	69	68		33	0	
20	0									
21							0			
22										
23										
28	65									
29										A
30	0									
31	1									
32							1			
33									1	
34										
35										
22										
24	67									
25	100									
26										d
27										
30	1									
31	2									
32							2			
33									2	
34										
36										

Question	Answer	Marks
3(a)	**1 mark** for each error **and** correction • Line 02 should be +1 not −1 // `PC ← [PC] + 1` • Line 03 should be double brackets around `MAR` // `MDR ← [[MAR]]` • Line 04 should be `MDR` not `MAR` // `CIR ← [MDR]`	**3**
3(b)	**1 mark** for each group to **max. 2** • Data movement • Arithmetic operations • (Unconditional and conditional) jump instructions • Compare instructions • Modes of addressing	**2**
3(c)	**1 mark** per bullet • Storing 0 in 401 (line 51) • Loading memory location 300, value 2 to ACC (line 52) • Adding 64 to ACC to give 66 (line 55) • Outputting B (line 56) • Load 0 (line 57), increment ACC (line 58) and store 1 in 401 (line 59) • Incrementing IX (line 60) • Loading 5 (line 52), adding 64 (line 55), outputting E (line 56) loading 1 (line 57), incrementing ACC (line 58), storing 2 in 401 (line 59) and incrementing IX (line 60) • Load 0 (line 52) and end	**8**

Instruction address	ACC	Memory address						IX	OUTPUT	
		300	301	302	303	400	401			
		2	5	0	4	64		0		
50	0									
51							0			[1]
52	2									[1]
53										
54										
55	66									[1]
56									B	[1]
57	0									[1]
58	1									
59							1			
60								1		[1]
61										[1]
52	5									
53										
54										
55	69									
56									E	
57	1									
58	2									
59							2			
60								2		
61										
52	0									[1]
53										
54										
62										

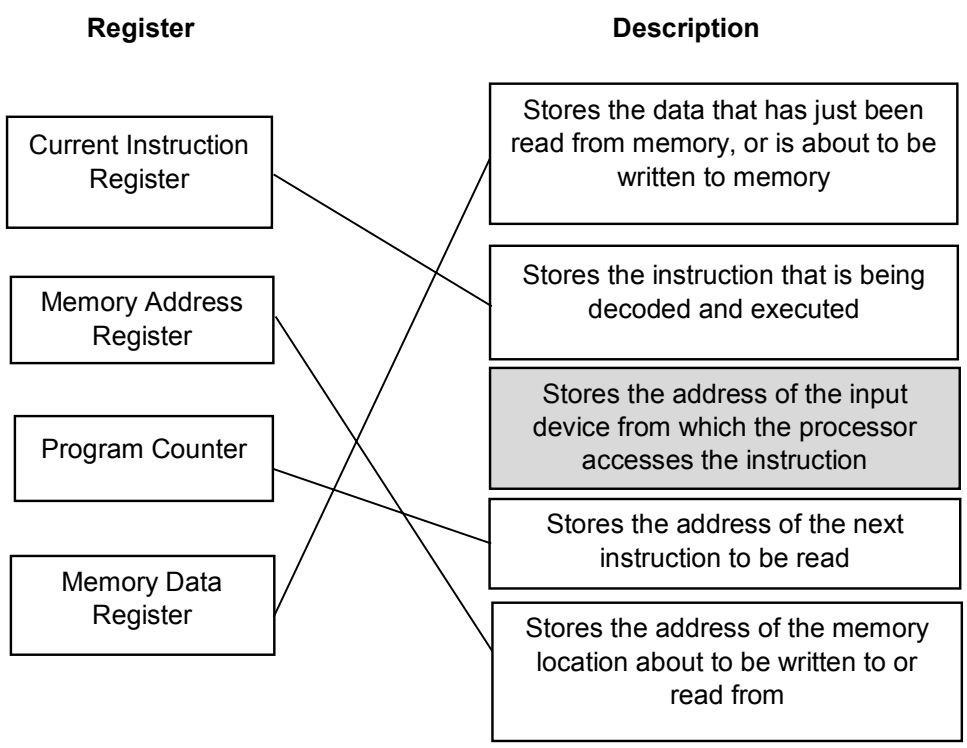
Question	Answer	Marks
3(d)(i)	**1 mark** for correct answer 0100 0001	**1**
3(d)(ii)	**1 mark** for correct answer 41	**1**
3(d)(iii)	**1 mark** for correct answer 0044	**1**

Question	Answer	Marks
4(a)	**1 mark** per bullet • LDM #300 The (denary) number 300 is loaded (into the register) • LDD 300 The contents of address 300 are loaded (into the register)	**2**
4(b)	**1 mark** for JPN correct **1 mark** for both of ADD and DEC correct **1** mark for LDR correct	**3**

	Description	Jump instruction	Arithmetic operation	Data movement
`LDR #3`	Load the number 3 to the Index Register			✓
`ADD #2`	Add 2 to the Accumulator		✓	
`JPN 22`	Move to the instruction at address 22	✓		
`DEC ACC`	Subtract 1 from the Accumulator		✓	

Question	Answer	Marks
4(c)(i)	**1 mark** for hardware interrupt **1 mark** for software interrupt For example: Hardware interrupt • Printer out of paper • No CD in drive Software interrupt • A running program needs input • Runtime error, e.g. division by zero	**2**
4(c)(ii)	**1 mark** per bullet to **max 5** • At the start / end of each fetch-execute cycle the processor checks for interrupt(s) • Check if an interrupt flag is set // Check if bit set in interrupt register • Processor identifies source of interrupt • Processor checks priority of interrupt • If interrupt priority is high enough // Lower priority interrupts are disabled • Processor saves current contents of registers // saves current job on stack • Processor calls interrupt handler / Interrupt Service Routine (ISR) • Address of ISR is loaded into Program Counter (PC) • When servicing of interrupt complete, processor restores registers // job from stack is restored • Lower priority interrupts are re-enabled • Processor continues with next F–E cycle	**5**

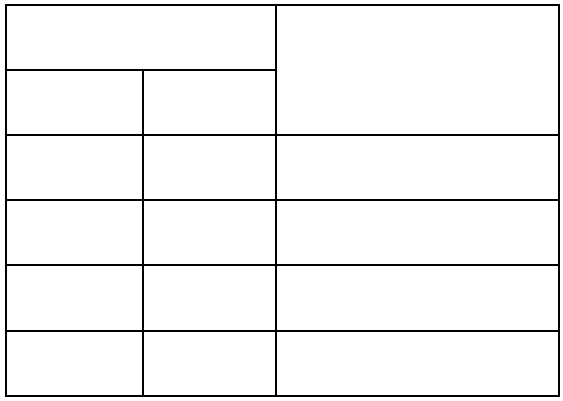
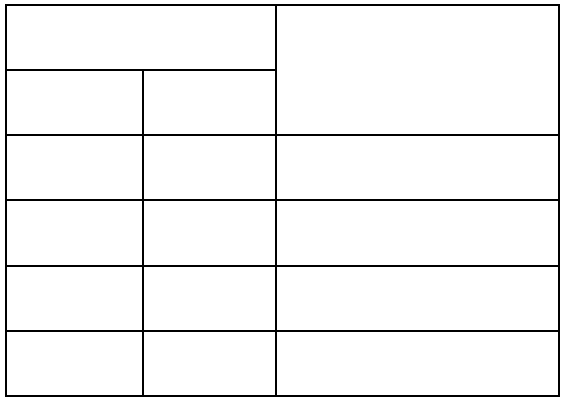
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7</td><td>1 mark for each correct addressing mode
<table>
<tr><th>Addressing mode</th><th>Description</th></tr>
<tr><td>Relative</td><td>Form the address by adding the given number to a base address. Load the contents of the calculated address to the Accumulator (ACC).</td></tr>
<tr><td>Indirect</td><td>Load the contents of the address held at the given address to ACC.</td></tr>
<tr><td>Direct</td><td>Load the contents of the given address to ACC.</td></tr>
<tr><td>Indexed</td><td>Form the address from the given address + the contents of the Index Register. Load the contents of the calculated address to ACC.</td></tr>
<tr><td>Immediate</td><td>Load the given value directly to ACC.</td></tr>
</table></td><td>5</td></tr>
</table>

Question	Answer	Marks
1(a)	**1 mark** for each correct line 	**4**
1(b)(i)	**1 mark** for naming, **1 mark** for purpose for each bus • Data bus • Carries data between the processor and memory / carries data that is currently being processed. • Control bus • Transmits signals between the control unit and the other components	**4**
1(b)(ii)	Significant increase in the number of directly addressed memory locations // increases the number of directly addressable memory locations from 2^{16} to2^{32}	**1**
1(c)	**1 mark** for each correctly inserted term A **macro** is a sequence of instructions that are given an identifier. These instructions may need to be executed several times. A **directive** is an instruction that tells the assembler to do something. It is not a program instruction. The processor's instruction set can be put into several groups. One of these groups is **data movement** // **input and output** // **arithmetic operations** // **jump instructions** // **compare instructions** // **modes of addressing**.	**3**

Question	Answer	Marks
5(a)	**1 mark** for **one** letter in the correct place, **2 marks** for all **three** correct 2 B 4 A 5 C	**2**
5(b)(i)	11000010	**1**
5(b)(ii)	CD	**1**
5(b)(iii)	**1 mark** per bullet point to **max 2** • The maximum range for an 8-bit two's complement binary number is −128 to +127 • … 200 is outside of the maximum range	**2**

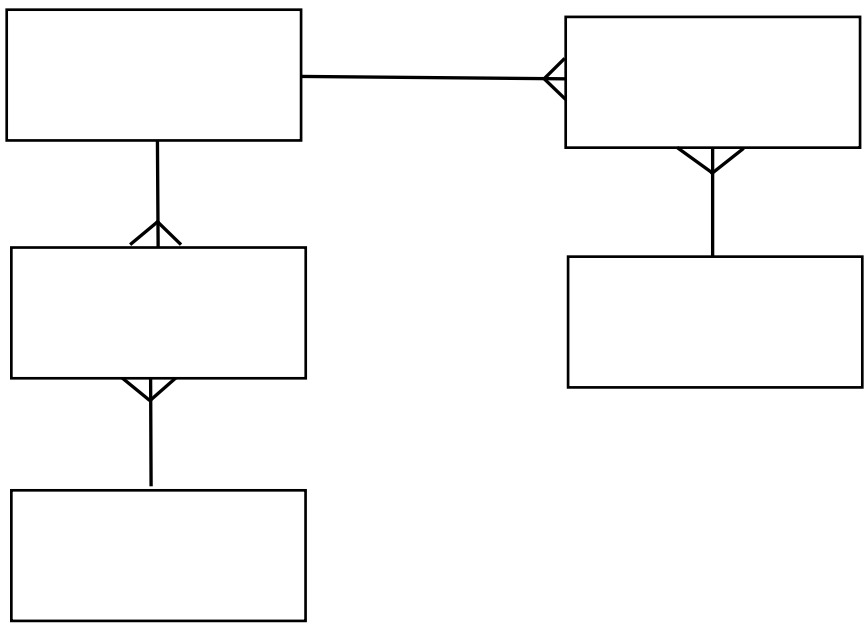
Question	Answer	Marks
5(c)	**1 mark** for each highlighted section block	**6**

Instruction address	ACC	Memory address			
		100	101	102	103
		1	2	3	0
20	0				
21					
22					
23	**1**				
24	**3**				
25		**3**			
26	**0**				
27	**1**				
28					**1**
29					
20	**1**				
21					
22					
23	**3**				
24	**5**				
25		**5**			
26	**1**				
27	**2**				
28					**2**
29					
20	**2**				
21					
22					
30					

Question	Answer	Marks
6(a)(i)	**1 mark** for each correct answer **A**: The number 193 **B**: The data in memory location 193 **C**: The data in the memory location found by adding the contents of the IX to 193	**3**
6(a)(ii)	**1 mark** each correct answer • Indirect • Relative	**2**

Question	Answer	Marks
6(b)	**1 mark** for correctly naming register, **1 mark** for appropriate role • Program counter // PC • Stores the address of the next instruction to be fetched • Memory address register // MAR • Stores the address where data/instruction is to be read from or saved to • Memory data register // MDR • Stores data that is about to be written to memory // Stores data that has just been read from memory • Current instruction register // CIR • Stores the instruction that is currently being decoded/executed	**4**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)</td><td>1 mark for tick in correct position

<table>
<tr><th>Statement</th><th>Tick (✓)</th></tr>
<tr><td>Memory location 204 contains <code>400</code></td><td></td></tr>
<tr><td>Memory location 204 contains <code>41</code></td><td></td></tr>
<tr><td>Memory location 204 contains <code>231</code></td><td></td></tr>
<tr><td>Memory location 204 contains <code>29</code></td><td>✓</td></tr>
</table></td><td>1</td></tr>
<tr><td>4(b)</td><td>1 mark for tick in correct position

<table>
<tr><th>Statement</th><th>Tick (✓)</th></tr>
<tr><td>Memory location 120 contains <code>135</code></td><td></td></tr>
<tr><td>Memory location 120 contains <code>118</code></td><td>✓</td></tr>
<tr><td>Memory location 120 contains <code>0</code></td><td></td></tr>
<tr><td>Memory location 120 contains <code>16</code></td><td></td></tr>
</table></td><td>1</td></tr>
<tr><td>4(c)</td><td>1 mark for tick in correct position

<table>
<tr><th>Statement</th><th>Tick (✓)</th></tr>
<tr><td>Memory location 205 contains <code>607</code></td><td></td></tr>
<tr><td>Memory location 205 contains <code>601</code></td><td></td></tr>
<tr><td>Memory location 205 contains <code>603</code></td><td></td></tr>
<tr><td>Memory location 205 contains <code>606</code></td><td>✓</td></tr>
</table></td><td>1</td></tr>
<tr><td>4(d)</td><td>1 mark per correct mode

• Indexed
• Relative</td><td>2</td></tr>
</table>

Question	Answer	Marks
4(e)	**1 mark** for correct ticks in pairs of rows (shaded)	**3**

Assembly language instruction	Arithmetic	Data movement	Jump instruction	Input and output of data
`STO 120`		✓		
`JPE 200`			✓	
`ADD 3`	✓			
`LDD 20`		✓		
`INC ACC`	✓			
`OUT`				✓

Question	Answer	Marks
3(a)	**1 mark** for each correct term **Stored** **Memory** **Fetches** **Decodes** **Executes** **Registers** The Von Neumann model uses the **stored** program concept. The program is a series of instructions that are saved in **memory**. The processor **fetches** each instruction, **decodes** it and then **executes** it. The processor uses several **registers** to store the data and instructions from the program because they can be accessed faster than main memory.	**6**

Question	Answer	Marks
3(b)(i)	**1 mark** for each set of shaded rows	**5**

Instruction address	ACC	Memory address			Output
		80	81	82	
		10	2	0	
50	10				
51	20				
52		20			
53	0				
54	1				
55				1	
56					
57					
50	20				
51	40				
52		40			
53	1				
54	2				
55				2	
56					
57					
58	40				
59					(
60					

Question	Answer	Marks
3(b)(ii)	**1 mark** for each correct column.	**3**

Instruction Group	Assembly language instruction		
	`STO 80`	`JPN 50`	`INC ACC`
Input and output of data			
Data movement	✓		
Arithmetic operations			✓
Unconditional and conditional jump instructions		✓	
Compare instructions			

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)(i)</td><td>1 mark for each correct row

<table>
<tr><th>Letter</th><th>Action</th><th>Register transfer notation</th></tr>
<tr><td>A</td><td>The Memory Address Register (MAR) stores an address. The contents of this stored address are copied to the Memory Data Register (MDR).</td><td>MDR ← [[MAR]]</td></tr>
<tr><td>B</td><td>The contents of the Program Counter (PC) are copied to the Memory Address Register (MAR).</td><td>MAR ← [PC]</td></tr>
<tr><td>C</td><td>The contents of the Memory Data Register (MDR) are copied to the Current Instruction Register (CIR).</td><td>CIR ← [MDR]</td></tr>
<tr><td>D</td><td>The contents of the Program Counter (PC) are incremented.</td><td>PC ← [PC] + 1</td></tr>
</table></td><td>4</td></tr>
<tr><td>5(a)(ii)</td><td>1 mark for B, A, C in order

1 mark for D at any point after B</td><td>2</td></tr>
<tr><td>5(b)</td><td>1 mark for the first three rows correct 1 mark for the last row correct

<table>
<tr><th>Statement</th><th>Address bus</th><th>Control bus</th><th>Data bus</th></tr>
<tr><td>Receives data from the MAR</td><td>✓</td><td></td><td></td></tr>
<tr><td>Carries an address or an instruction or a value</td><td></td><td></td><td>✓</td></tr>
<tr><td>Transmits timing signals to components</td><td></td><td>✓</td><td></td></tr>
<tr><td>Bidirectional</td><td></td><td>✓</td><td>✓</td></tr>
</table></td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(c)(i)</td><td>1 mark for each correct example, ignore operand

Arithmetic:
• ADD // INC

Data movement
• STO // LDD // LDI // LDM</td><td>2</td></tr>
<tr><td>5(c)(ii)</td><td>1 mark for each correct shaded section

<table>
<tr><th rowspan="2">Instruction address</th><th rowspan="2">ACC</th><th colspan="4">Memory address</th></tr>
<tr><th>900</th><th>901</th><th>902</th><th>903</th></tr>
<tr><td></td><td></td><td></td><td></td><td>0</td><td>0</td></tr>
<tr><td>500</td><td>2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>501</td><td></td><td></td><td>2</td><td></td><td></td></tr>
<tr><td>502</td><td>10</td><td></td><td></td><td></td><td></td></tr>
<tr><td>503</td><td></td><td>10</td><td></td><td></td><td></td></tr>
<tr><td>504</td><td>10</td><td></td><td></td><td></td><td></td></tr>
<tr><td>505</td><td></td><td></td><td></td><td>10</td><td></td></tr>
<tr><td>506</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>507</td><td>1</td><td></td><td></td><td></td><td></td></tr>
<tr><td>508</td><td></td><td></td><td></td><td></td><td>1</td></tr>
<tr><td>509</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>510</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>502</td><td>3</td><td></td><td></td><td></td><td></td></tr>
<tr><td>503</td><td></td><td>3</td><td></td><td></td><td></td></tr>
<tr><td>504</td><td>13</td><td></td><td></td><td></td><td></td></tr>
<tr><td>505</td><td></td><td></td><td></td><td>13</td><td></td></tr>
<tr><td>506</td><td>1</td><td></td><td></td><td></td><td></td></tr>
<tr><td>507</td><td>2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>508</td><td></td><td></td><td></td><td></td><td>2</td></tr>
<tr><td>509</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>511</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>6</td></tr>
<tr><td>5(d)(i)</td><td>202</td><td>1</td></tr>
<tr><td>5(d)(ii)</td><td>–54</td><td>1</td></tr>
<tr><td>5(d)(iii)</td><td>Both nibbles are binary numbers representing denary numbers greater than 9</td><td>1</td></tr>
</table>