Chapter 4
19-20

5 A simple program written in assembly language is translated using a two-pass assembler.

(a) The table contains some of the tasks performed by a two-pass assembler.

Tick (✓) **one** box in each row to indicate whether the task is performed at the first or second pass. The first row has been completed for you.

Task	First pass	Second pass
Creation of symbol table	✓	
Expansion of macros		
Generation of object code		
Removal of comments		

[2]

(b) The processor's instruction set can be grouped according to their function. For example, one group is modes of addressing.

Identify **two** other groups of instructions.

1 .....

.....

2 .....

.....

[2]

(c) The table shows assembly language instructions for a processor which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the denary number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given 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 denary number n to IX.
STO	<address>	Store contents of ACC at the given address.
ADD	<address>	Add the contents of the given address to ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
CMP	#n	Compare contents of ACC with denary number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
JMP	<address>	Jump to the given address.
OUT		Output to screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

The current contents of the main memory, Index Register (IX) and selected values from the ASCII character set are:

Address	Instruction
20	LDM #0
21	STO 300
22	CMP #0
23	JPE 28
24	LDX 100
25	ADD 301
26	OUT
27	JMP 30
28	LDX 100
29	OUT
30	LDD 300
31	INC ACC
32	STO 300
33	INC IX
34	CMP #2
35	JPN 22
36	END
...	
100	65
101	67
102	69
103	69
104	68
...	
300	
301	33
IX	0

ASCII code table (Selected codes only)

ASCII Code	Character
65	A
66	B
67	C
68	D
69	E
97	a
98	b
99	c
100	d
101	e

Trace the program currently in memory using the following trace table. The first instruction has been completed for you.

Instruction address	ACC	Memory address							IX	OUTPUT
		100	101	102	103	104	300	301		
		65	67	69	69	68		33	0	
20	0									

[8]

3 The fetch-execute cycle is shown in register transfer notation.

```
01     MAR ← [PC]

02     PC ← [PC] - 1

03     MDR ← [MAR]

04     CIR ← [MAR]
```

(a) There are **three** errors in the fetch-execute cycle shown.

Identify the line number of each error and give the correction.

Line number .....

Correction .....

Line number .....

Correction .....

Line number .....

Correction .....
[3]

(b) A processor's instruction set can be grouped according to their function. For example, one group is the input and output of data.

Identify **two** other groups of instructions.

1 .....

.....

2 .....

.....
[2]

(c) The following table shows assembly language instructions for a processor which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the denary number n to ACC
LDD	<address>	Direct addressing. Load the contents of the location at the given 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 denary number n to IX
STO	<address>	Store contents of ACC at the given address
ADD	<address>	Add the contents of the given address to ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
CMP	#n	Compare contents of ACC with denary number n
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
JMP	<address>	Jump to the given address
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system

The current contents of the main memory, Index Register (IX) and selected values from the ASCII character set are:

Address	Instruction
50	LDM #0
51	STO 401
52	LDX 300
53	CMP #0
54	JPE 62
55	ADD 400
56	OUT
57	LDD 401
58	INC ACC
59	STO 401
60	INC IX
61	JMP 52
62	END
...	
300	2
301	5
302	0
303	4
...	
400	64
401	
IX	0

ASCII code table (Selected codes only)

ASCII code	Character
65	A
66	B
67	C
68	D
69	E

Trace the program currently in memory using the following trace table.
The first instruction has been completed for you.

Instruction address	ACC	Memory address						IX	OUTPUT
		300	301	302	303	400	401		
		2	5	0	4	64		0	
50	0								

[8]

(d) The ASCII character code for ‘A’ is 65 in denary.

(i) Convert the denary ASCII character code for ‘A’ into 8-bit binary.

[1]

(ii) Convert the denary ASCII character code for ‘A’ into hexadecimal.

..... [1]

(iii) The Unicode character code for ‘G’ is 0047 in hexadecimal.

State, in hexadecimal, the Unicode character code for ‘D’.

..... [1]

4 A program is written in assembly language.

(a) The op codes `LDM` and `LDD` are used to load a register. The op code `LDM` uses immediate addressing, and the op code `LDD` uses direct addressing.

Describe what happens when the following instructions are run.

```
LDM #300
```

.....

.....

```
LDD 300
```

.....

.....

[2]

(b) Assembly language instructions can be grouped by their purpose.

The following table shows four assembly language instructions.

Tick (✓) **one** box in each row to indicate the group each instruction belongs to.

Instruction	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			

[3]

(c) The processor handles interrupts within the fetch-execute cycle.

(i) Give **one** example of a hardware interrupt and **one** example of a software interrupt.

Hardware .....

.....

Software .....

.....
[2]

(ii) Explain how the processor handles an interrupt.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

7 The following table has descriptions of modes of addressing.

Complete the table by writing the name of the addressing mode for each description.

Addressing mode	Description
	Form the address by adding the given number to a base address. Load the contents of the calculated address to the Accumulator (ACC).
	Load the contents of the address held at the given address to ACC.
	Load the contents of the given address to ACC.
	Form the address from the given address + the contents of the Index Register. Load the contents of the calculated address to ACC.
	Load the given value directly to ACC.

[5]

1 Von Neumann is an example of a computer architecture.

(a) The diagram has registers used in Von Neumann architecture on the left and descriptions on the right.

Draw **one** line to match each register with its correct description.

Register	Description
Current Instruction Register	Stores the data that has just been read from memory, or is about to be written to memory
Memory Address Register	Stores the instruction that is being decoded and executed
Program Counter	Stores the address of the input device from which the processor accesses the instruction
Memory Data Register	Stores the address of the next instruction to be read
	Stores the address of the memory location about to be written to or read from

[4]

(b) Many components of the computer system transfer data between them using buses. One example of a bus is an address bus.

(i) Name **two** other buses that exist within a computer and give the purpose of each.

Bus 1 .....

Purpose .....

.....

.....

Bus 2 .....

Purpose .....

.....

.....
[4]

(ii) State the benefit of increasing the address bus width from 16 bits to 32 bits.

.....

..... [1]

(c) The following statements describe features of a low-level language.

Complete the statements by writing the appropriate terms in the spaces.

A ..... is a sequence of instructions that are given an identifier. These instructions may need to be executed several times.

A ..... 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 ..... .

[3]

5 **(a)** The steps 1 to 6 describe the first pass of a two-pass assembler.

The following three statements are used to complete the sequence of steps.

A	If it is already in the symbol table, it checks to see if the absolute address is known
B	When it meets a symbolic address, it checks to see if it is already in the symbol table
C	If it is known, it is entered

Write one of the letters **A**, **B** or **C** in the appropriate step to complete the sequence.

1. The assembler reads the assembly language instructions
2. .....
3. If it is not, it adds it to the symbol table
4. .....
5. .....
6. If it is not known, it is marked as unknown.

[2]

(b) The assembler translates assembly code into machine code.

The table shows the denary values for three assembler op codes.

Op code	Denary value
`LDD`	194
`ADD`	200
`STO`	205

(i) Convert the denary value for the op code `LDD` into 8-bit binary.

[1]

(ii) Convert the denary value for the op code `STO` into hexadecimal.

..... [1]

(iii) State why the denary value for the op code `ADD` cannot be represented in 8-bit two's complement form. Justify your answer.

.....

.....

.....

..... [2]

(c) The table shows part of the instruction set for a processor. The processor has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the denary number n to ACC
LDD	<address>	Direct addressing. Load the contents of the location at the given 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 denary number n to IX
STO	<address>	Store contents of ACC at the given address
ADD	<address>	Add the contents of the given address to ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
CMP	<address>	Compare contents of the address given with the contents of ACC
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
JMP	<address>	Jump to the given address
OUT		Output to screen the character whose ASCII value is stored in ACC
END		Return control to the operating system

Complete the trace table for the following assembly language program. The first instruction has been completed for you.

Address	Instruction
20	LDD 103
21	CMP 101
22	JPE 30
23	LDD 100
24	ADD 101
25	STO 100
26	LDD 103
27	INC ACC
28	STO 103
29	JMP 20
30	END
...	
100	1
101	2
102	3
103	0

Instruction address	ACC	Memory address			
		100	101	102	103
		1	2	3	0
20	0				

[6]

6 A processor has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

(a) The table gives **three** assembly language instructions for loading data into the ACC. It also identifies the addressing mode used for each instruction.

	Instruction	Addressing mode
A	`LDM #193`	Immediate
B	`LDD 193`	Direct
C	`LDX 193`	Indexed

(i) State the contents of the Accumulator after each of the instructions **A**, **B** and **C** are run.

A .....

.....

B .....

.....

C .....

.....

[3]

(ii) Name **two** other addressing modes.

1 .....

2 .....

[2]

(b) The ACC is a general purpose register. The IX is a special purpose register.

Identify **two** other special purpose registers used in the fetch-execute cycle **and** describe their role in the cycle.

Register 1 .....

Role .....

.....

.....

Register 2 .....

Role .....

.....

.....

[4]

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

Instruction		Explanation
Op code	**Operand**	
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDM	#n	Immediate addressing. Load the denary number n 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.
CMP	<address>	Compare the contents of ACC with <address>.
STO	<address>	Store contents of ACC at the given address.
ADD	<address>	Add the contents of the given address to ACC.
SUB	<address>	Subtract the contents of the given address from the contents of ACC.
OUT		Output to screen the character whose ASCII value is stored in ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
END		Return control to the operating system.

(a) The current contents of the main memory are:

Address	Instruction
100	LDD 200
101	ADD 201
102	ADD 202
103	SUB 203
104	STO 204
105	END
…	
200	10
201	20
202	5
203	6
204	
205	

Tick (✓) **one** box to indicate which **one** of the following statements is **true** after program execution.

Statements	Tick (✓)
Memory location 204 contains `400`	
Memory location 204 contains `41`	
Memory location 204 contains `231`	
Memory location 204 contains `29`	

[1]

(b) The current contents of the main memory are:

Address	Instruction
100	`LDM #120`
101	`ADD 121`
102	`SUB 122`
103	`STO 120`
104	`END`
…	
120	`10`
121	`2`
122	`4`
123	`6`
124	`8`
125	`10`

Tick (✓) **one** box to indicate which **one** of the following statements is **true** after program execution.

Statement	Tick (✓)
Memory location 120 contains `135`	
Memory location 120 contains `118`	
Memory location 120 contains `0`	
Memory location 120 contains `16`	

[1]

(c) The current contents of the main memory are:

Address	Instruction
150	`LDI 200`
151	`ADD 200`
152	`ADD 201`
153	`STO 205`
154	`END`
…	
200	`202`
201	`203`
202	`201`
203	`200`
204	
205	

Tick (✓) **one** box to indicate which **one** of the following statements is **true** after program execution.

Statement	Tick (✓)
Memory location 205 contains `607`	
Memory location 205 contains `601`	
Memory location 205 contains `603`	
Memory location 205 contains `606`	

[1]

(d) Identify **two** modes of addressing that are **not** used in **parts (a)**, **(b)** or **(c)**.

1 .....

2 .....

[2]

(e) Assembly language instructions can be put into groups.

Tick (✓) **one** box on each row to indicate the appropriate instruction group for each assembly language instruction.

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

[3]

2 One method of compressing a file is run-length encoding (RLE).

(a) Describe, using an example, how a **text file** is compressed using RLE.

.....

.....

.....

.....

.....

..... [3]

(b) Explain why run-length encoding will sometimes increase the size of a text file.

.....

.....

.....

..... [2]

3 **(a)** Complete the following statements about CPU architecture by filling in the missing terms.

The Von Neumann model for a computer system uses the ..... program concept.

A program is a series of instructions that are saved in ..... .

The processor ..... each instruction, ... it and then

..... it.

The processor uses several ..... to store the data and instructions from

the program because they can be accessed faster than main memory.

[6]

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

Instruction		Explanation
Op code	**Operand**	
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDM	#n	Immediate addressing. Load the denary number n 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.
STO	<address>	Store contents of ACC at the given address.
ADD	<address>	Add the contents of the given address to ACC.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
OUT		Output to screen the character whose ASCII value is stored in ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
END		Return control to the operating system.

(i) The current contents of the main memory are:

Address	Instruction
50	LDD 80
51	ADD 80
52	STO 80
53	LDD 82
54	INC ACC
55	STO 82
56	CMP 81
57	JPN 50
58	LDD 80
59	OUT
60	END
…	
80	10
81	2
82	0

ASCII code table (Selected codes only)

ASCII Code	Character
38	&
39	‘
40	(
41	)
42	*

Trace the program currently in memory using the following trace table.
The first instruction has been completed for you.

Instruction address	ACC	Memory address			Output
		80	81	82	
		10	2	0	
50	10				

[5]

(ii) Assembly language instructions can be put into groups.

Tick (✓) **one** box in each column to identify the appropriate instruction group for each of the three assembly language instructions.

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			

[3]

4 **(a)** Complete the truth table for the logic expression:

X = ((**A** NOR **B**) AND (**C** XOR **A**)) OR **B**

A	**B**	**C**	**Working space**	**X**
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

[4]

(b) Describe the difference between the operation of an **AND** gate and a **NAND** gate.

.....

.....

.....

..... [2]

5 The fetch-execute cycle is used when a computer processor runs a program.

(a) **(i)** Complete the table by writing the register transfer notation for each of the descriptions.

Letter	Description	Register transfer notation
A	The Memory Address Register (MAR) stores an address. The contents of this stored address are copied to the Memory Data Register (MDR).	
B	The contents of the Program Counter (PC) are copied to the Memory Address Register (MAR).	
C	The contents of the Memory Data Register (MDR) are copied to the Current Instruction Register (CIR).	
D	The contents of the Program Counter (PC) are incremented.	

[4]

(ii) Write one of the letters **A**, **B**, **C** or **D** (from the table above) on each row (1 to 4), to show the correct order of the fetch-execute cycle.

1 .....

2 .....

3 .....

4 .....

[2]

(b) Buses are used to transfer data between various components of the computer system.

Tick (✓) **one or more** boxes on each row to identify the bus(es) each statement describes.

Statement	Address bus	Control bus	Data bus
Receives data from the MAR			
Carries an address or an instruction or a value			
Transmits timing signals to components			
Bidirectional			

[2]

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

Instruction		Explanation
Op code	**Operand**	
INV		Input a denary value from the keyboard and store it in ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDM	#n	Immediate addressing. Load the denary number n 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.
ADD	<address>	Add the contents of the given address to ACC.
OUT		Output to screen the character whose ASCII value is stored in ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
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.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
STO	<address>	Store contents of ACC at the given address.
END		Return control to the operating system.

(i) The assembly language instructions are grouped according to their function.

Write **one** example of an op code from the table of instructions for each of the following groups.

Arithmetic .....

Data movement .....

[2]

(ii) The current contents of the main memory are:

Address	Instruction
500	INV
501	STO 901
502	INV
503	STO 900
504	ADD 902
505	STO 902
506	LDD 903
507	INC ACC
508	STO 903
509	CMP 901
510	JPN 502
511	END
...	
900	
901	
902	0
903	0

Trace the program currently in memory using the following trace table when the values 2, 10 and 3 are input.

The first instruction has been completed for you.

Instruction address	ACC	Memory address			
		900	901	902	903
				0	0
500	2				

[6]

(d) The current contents of a general-purpose register **X** are:

X	1	1	0	0	1	0	1	0

(i) The contents of **X** represent an unsigned binary integer.

Convert the contents of **X** into denary.

..... [1]

(ii) The contents of **X** represent a two's complement binary integer.

Convert the contents of **X** into denary.

..... [1]

(iii) State why the binary number in **X** cannot represent a Binary Coded Decimal (BCD).

.....

..... [1]