Assembly Language Programming

Question 1

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]

Question 2

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]

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				

Question 3

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

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

Question 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

Question 5

Biyu is writing a computer program in a high-level language.

(a) Biyu uses a language translator.

(i) State the purpose of a language translator.

....

....

[1]

(ii) Biyu uses an interpreter. State two benefits of Biyu using an interpreter instead of a compiler while writing the program.

1 ....

....

2 ....

....

[2]

(iii) Name a translator other than an interpreter and a compiler.

....

[1]

(b) Biyu uses library files in the program. Explain why software is often developed using library files.

....

....

....

.... [2]

Question 6

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

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]

Question 7

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.

Question 8

(a) (i) State what is meant by direct addressing and indirect addressing.

Direct addressing ....

....

Indirect addressing

....

....

[2]

(ii) Explain how the instruction ADD 20 can be interpreted as either direct or indirect addressing.

Direct addressing

....

....

Indirect addressing

....

.... [2]

(b) The assembly language instructions in the following table use either symbolic addressing or absolute addressing.

Tick (✓) **one** box in each row to indicate whether the instruction uses symbolic or absolute addressing.

Instruction	Symbolic	Absolute
`ADD 90`		
`CMP found`		
`STO 20`		

[2]

(c) The current contents of a general purpose register (X) are:

X	1	0	1	1	1	0	1	0

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

Convert the value in X into denary.

....[1]

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

Convert the value in X into hexadecimal.

....[1]

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

Convert the value in X into denary.

....[1]

Address	Instruction
70	LDX 200
71	OUT
72	STO 203
73	LDD 204
74	INC ACC
75	STO 204
76	INC IX
77	LDX 200
78	CMP 203
79	JPN 81
80	OUT
81	LDD 204
82	CMP 205
83	JPN 74
84	END
…	
200	130
201	133
202	130
203	0
204	0
205	2

IX	0

ASCII code table (selected codes only)

ASCII code	Character
127	?
128	!
129	"
130	*
131	$
132	&
133	%
134	/

Instruction set

Instruction		Explanation
Op code	Operand	
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 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).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
CMP	<address>	Compare contents of ACC with 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.
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.

Instruction address	ACC	Memory address						IX	OUTPUT
		200	201	202	203	204	205		
70	130	130	133	130	0	0	2	0	

[8]

Question 9

(a) (i) State what is meant by absolute addressing and symbolic addressing.

Absolute addressing

....

....

Symbolic addressing

....

.... [2]

(ii) Give an example of an ADD instruction using both absolute addressing and symbolic addressing.

Absolute addressing ....

Symbolic addressing .... [2]

(b) (i) State what is meant by indexed addressing and immediate addressing.

Indexed addressing

....

....

Immediate addressing

....

.... [2]

(ii) Give an example of an instruction that uses:

Indexed addressing ....

Immediate addressing ....[2]

(c) The current contents of a general purpose register (X) are:

X	1	1	0	0	0	0	0	1

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

Convert the value in X into denary.

.... [1]

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

Convert the value in X into hexadecimal.

.... [1]

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

Convert the value in X into denary.

.... [1]

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

Address	Instruction
40	LDD 100
41	CMP 104
42	JPE 54
43	LDX 100
44	CMP 105
45	JPN 47
46	OUT
47	LDD 100
48	DEC ACC
49	STO 100
50	INC IX
51	JMP 41
52	
53	
54	END
...	
100	2
101	302
102	303
103	303
104	0
105	303

IX	1

ASCII code table (selected codes only)

ASCII code	Character
300	/
301	*
302	-
303	+
304	^
305	=

Instruction address	ACC	Memory address						IX	OUTPUT
		100	101	102	103	104	105		
		2	302	303	303	0	303	1	
40									

[7]

Question 10

(a) State what is meant by relative addressing and indexed addressing.

Relative addressing ....

....

....

Indexed addressing ....

....

.... [2]

(b) The current contents of a general purpose register (X) are:

X	1	1	1	1	0	0	1	0

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

Convert the value in X into denary.

....[1]

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

Convert the value in X into hexadecimal.

....[1]

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

Convert the value in X into denary.

....[1]

(iv) Show the result on the general purpose register (X) after the following instruction is run.

`INC X`

[1]

6

Address	Instruction
20	LDD 96
21	CMP 97
22	JPE 32
23	LDX 86
24	CMP 98
25	JPN 27
26	OUT
27	LDD 96
28	INC ACC
29	STO 96
30	INC IX
31	JMP 21
32	END
…	
93	453
94	453
95	452
96	8
97	10
98	453

IX	8

ASCII code table (selected codes only)

ASCII code	Character
450	<
451	>
452	=
453	&
454	(
455	)

Instruction set

Instruction		Explanation
Op code	Operand	
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 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).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
CMP	<address>	Compare contents of ACC with contents of <address>.

Instruction address	ACC	Memory address						IX	OUTPUT
		93	94	95	96	97	98		
		453	453	452	8	10	453	8	
20									

Question 11

4 The following 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 (mnemonic)	**Operand**	**Op code (binary)**	
`LDM`	`#n`	0000 0001	Immediate addressing. Load the denary number `n` to ACC.
`LDD`	`<address>`	0000 0010	Direct addressing. Load the contents of the location at the given address to ACC.
`LDI`	`<address>`	0000 0101	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC.
`LDX`	`<address>`	0000 0110	Indexed addressing. Form the address from `<address>` + the contents of the Index Register (IX). Copy the contents of this calculated address to ACC.
`LDR`	`#n`	0000 0111	Immediate addressing. Load number `n` to IX.
`STO`	`<address>`	0000 1111	Store the contents of ACC at the given address.

The following diagram shows the contents of a section of main memory and the Index Register (IX).

(a) Show the contents of the Accumulator (ACC) after each instruction is executed.

IX	0	0	0	0	0	0	1	1

(i) `LDM #500`

ACC ....[1]

(ii) `LDD 500`

ACC ....[1]

(iii) `LDX 500`

ACC ....[1]

(iv) `LDI 500`

ACC ....[1]

Address	Main Memory contents
495	13
496	86
497	92
498	486
499	489
500	496
501	497
502	499
503	502

(b) Each machine code instruction is encoded as 16-bits (8-bit op code followed by an 8-bit operand).

Write the machine code for the following instructions:

`LDM #17`

`LDX #97`

[3]

(c) Using an 8-bit operand, state the maximum number of memory locations, in denary, that can be directly addressed.

....[1]

(d) Computer scientists often write binary representations in hexadecimal.

(i) Write the hexadecimal representation for this instruction:

0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	0

....[2]

(ii) A second instruction has been written in hexadecimal as:

`05 3F`

Write the equivalent assembly language instruction, with the operand in denary.

....[2]

Question 12

Instruction			
Op code (mnemonic)	Operand	Op code (binary)	Explanation
LDD	<address>	0001 0011	Direct addressing. Load the contents of the location at the given address to the Accumulator (ACC).
LDI	<address>	0001 0100	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	0001 0101	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDM	#n	0001 0010	Immediate addressing. Load the denary number n to ACC.
LDR	#n	0001 0110	Immediate addressing. Load denary number n to the Index Register (IX).
STO	<address>	0000 0111	Store the contents of ACC at the given address.

The following diagram shows the contents of a section of main memory and the Index Register (IX).

(a) Show the contents of the Accumulator (ACC) after each instruction is executed.

IX	0	0	0	0	0	1	1	0

Address	Main memory contents
350	
351	86
352	
353	
354	
355	351
356	
357	22
358	

(i) `LDD 355`

ACC .... [1]

(ii) `LDM #355`

ACC .... [1]

(iii) `LDX 351`

ACC .... [1]

(iv) `LDI 355`

ACC .... [1]

(b) Each machine code instruction is encoded as 16 bits (8-bit op code followed by an 8-bit operand).

Write the machine code for these instructions:

`LDM #67`

`LDX #7`

[3]

(c) Computer scientists often write binary representations in hexadecimal.

(i) Write the hexadecimal representation for the following instruction.

0	0	0	1	0	1	0	0	0	1	0	1	1	1	1	0

....[2]

(ii) A second instruction has been written in hexadecimal as:

`16 4D`

Write the assembly language for this instruction with the operand in denary.

....[2]

Question 13

Label	Instruction	
StartProg:	LDV	#CountDown
	CMP	Num1
	JNE	CarryOn
	JMP	Finish
CarryOn:	OUTCH	
	LDD	CountDown
	DEC	
	STO	CountDown
	JMP	StartProg
Finish:	LDM	#88
	OUTCH	
	END	
CountDown:	15	
	32	
	51	
	67	
Num1:	32	

ASCII code table (selected codes only)				
<Space>	3	B	C	X
32	51	66	67	88

Trace table:

ACC	CountDown	OUTPUT
	15	
67		C
15		

[5]

(c) The program given in **part (b)** is to be translated using a two-pass assembler.

The program has been copied here for you. The program now starts with a directive which tells the assembler to load the first instruction of the program to address 100.

Label		
	ORG	#0100
StartProg:	LDV	#CountDown
	CMP	Num1
	JNE	CarryOn
	JMP	Finish
CarryOn:	OUTCH	
	LDD	CountDown
	DEC	
	STO	CountDown
	JMP	StartProg
Finish:	LDM	#88
	OUTCH	
	END	
CountDown:		15
		32
		51
		67
Num1:		32

On the first pass of the two-pass process, the assembler adds entries to a symbol table.

The following symbol table shows the first eleven entries, part way through the first pass.

The circular labels show the order in which the assembler made the entries to the symbol table.

Symbol table

Symbolic address		Absolute address			
`StartProg`	1	100	2		
`CountDown`	3	UNKNOWN	4		
`Num1`	5	UNKNOWN	6		
`CarryOn`	7	~~UNKNOWN~~	8	104	11
`Finish`	9	UNKNOWN	10		

Explain how the assembler made these entries to the symbol table.

....

....

....

....

....

....[3]

(d) The assembler software must then complete the second pass building up the executable file.

(i) Name the second table needed when the assembler software carries out the second pass.

....[1]

The following shows two of the program instructions in machine code.

Instruction	Machine code	
	Binary	**Hexadecimal**
`OUTCH`	`1100 0111`	`C7`
`JNE CarryOn`	**A**	**B**

Each of the numbers **A** and **B** represents the complete instruction in two bytes, one byte for the op code and one byte for the operand.

(ii) Use the following instruction set to write the numbers for **A** and **B**.

A (binary) ....

B (hexadecimal) ....

[3]

Instruction			Explanation
Op code (mnemonic)	Operand	Op code (binary)	
LDM	#n	1100 0001	Immediate addressing. Load number n to ACC.
LDD	<address>	1100 0010	Direct addressing. Load the contents of the given address to ACC.
LDV	#n	1100 0011	Relative addressing. Move to the address n locations from the address of the current instruction. Load the contents of this address to ACC.
STO	<address>	1100 0100	Store the contents of ACC at the given address.
DEC		1100 0101	Decrement the contents of ACC.
OUTCH		1100 0111	Output the character corresponding to the ASCII character code in ACC.
JNE	<address>	1110 0110	Following a compare instruction, jump to <address> if the compare was False.
JMP	<address>	1110 0011	(Unconditionally) jump to the given address.
CMP	#n	1110 0100	Compare the contents of ACC with number n.

Question 14

(b) Complete the trace table on the opposite page for the following assembly language program.

50	LDD 100
51	ADD 102
52	STO 103
53	LDX 100
54	ADD 100
55	CMP 101
56	JPE 58
57	JPN 59
58	OUT
59	INC IX
60	LDX 98
61	ADD 101
62	OUT
63	END
...	
100	20
101	100
102	1
103	0

IX (Index Register)	1

Selected values from the ASCII character set:

ASCII Code	118	119	120	121	122	123	124	125
Character	v	w	x	y	z	{	|	}

Trace table:

Instruction address	Working space	ACC	Memory address				IX	OUTPUT
			100	101	102	103		
			20	100	1	0	1	
50								
51								
52								
53								
54								
55								

[7]

Question 15

(a) The diagram shows the current contents of a section of main memory and the index register:

60	0011 0010
61	0101 1101
62	0000 0100
63	1111 1001
64	0101 0101
65	1101 1111
66	0000 1101
67	0100 1101
68	0100 0101
69	0100 0011
...	
1000	0110 1001

Index register:

0	0	0	0	1	0	0	0

(i) Show the contents of the Accumulator after the execution of the instruction:

`LDX 60`

Accumulator:

Show how you obtained your answer.

....

....

....

....[2]

(ii) Show the contents of the index register after the execution of the instruction:

`DEC IX`

Index register:

[1]

Question 16

The diagram shows the contents of the index register:

Index register:

1	1	0	0	1	1	0	1

(a) Show the contents of the index register after the execution of the instruction:

`INC IX`

Index register:

[1]

(b) Complete the trace table on the opposite page for the following assembly language program.

20	`LDX 90`
21	`DEC ACC`
22	`STO 90`
23	`INC IX`
24	`LDX 90`
25	`DEC ACC`
26	`CMP 90`
27	`JPE 29`
28	`JPN 31`
29	`ADD 90`
30	`OUT`
31	`ADD 93`
32	`STO 93`
33	`OUT`
34	`END`
:	
90	2
91	90
92	55
93	34

IX	2

Selected values from the ASCII character set:

ASCII Code	65	66	67	68	69	70	71	72
Character	A	B	C	D	E	F	G	H

Trace table:

Instruction	Working space	ACC	Memory address				IX	OUTPUT
			90	91	92	93		
			2	90	55	34	2	
20								
21								
22								
23								
24								
25								
26								

[7]

Question 17

The diagram shows the contents of the main memory:

Main memory

800	0110 0100
801	0111 1100
802	1001 0111
803	0111 0011
804	1001 0000
805	0011 1111
806	0000 1110
807	1110 1000
808	1000 1110
809	1100 0010
:	
2000	1011 0101

(a) **(i)** Show the contents of the Accumulator after execution of the instruction:

`LDD 802`

Accumulator:								

[1]

(ii) Show the contents of the Accumulator after execution of the instruction:

`LDX 800`

Index Register:	0	0	0	0	1	0	0	1
Accumulator:								

Explain how you arrived at your answer.

....

....

....

....[3]

(b) (i) Complete the trace table below for the following assembly language program. This program contains denary values.

100	LDD 800
101	ADD 801
102	STO 802
103	LDD 803
104	CMP 802
105	JPE 107
106	JPN 110
107	STO 802
108	OUT
109	JMP 112
110	LDD 801
111	OUT
112	END
⋮	
800	40
801	50
802	0
803	90

Selected values from the ASCII character set:

ASCII code	40	50	80	90	100
Character	(	2	P	Z	d

Trace table:

ACC	Memory address				OUTPUT
	800	801	802	803	
	40	50	0	90	

[4]

(ii) There is a redundant instruction in the code in part (b)(i).

State the address of this instruction.

....[1]

Question 18

The diagram shows the contents of a section of main memory:

Main memory

100	0000 0010
101	1001 0011
102	0111 0011
103	0110 1011
104	0111 1110
105	1011 0001
106	0110 1000
107	0100 1011
...	
200	1001 1110

(a) (i) Show the contents of the Accumulator after the execution of the instruction:

LDD 102

ACC:								

[1]

(ii) Show the contents of the Accumulator after the execution of the instruction:

LDX 101

IX:	0	0	0	0	0	1	0	0

ACC:								

Explain how you arrived at your answer.

....

....

....

....[2]

(iii) Show the contents of the Accumulator after the execution of the instruction:

LDI 103

ACC:								

Explain how you arrived at your answer.

....

....

....

....

....

.... [3]

(b) Complete the trace table below for the following assembly language program.

800	`LDD 810`
801	`INC ACC`
802	`STO 812`
803	`LDD 811`
804	`ADD 812`
805	`STO 813`
806	`END`
...	
810	`28`
811	`41`
812	`0`
813	`0`

Trace table:

ACC	Memory address			
	810	811	812	813
	28	41	0	0

[6]

Question 19

3 Five modes of addressing and five descriptions are shown below.

Draw a line to connect **each** mode of addressing to its correct description.

Mode of addressing	**Description**
direct	the operand is the address of the address of the value to be used
immediate	the operand is the address of the value to be used
indexed	the operand is the offset from the current address where the value to be used is stored
indirect	the operand plus the contents of the index register is the address of the value to be used
relative	the operand is the value to be used

[4]

Question 20

The diagram shows the contents of the memory.

Main memory

120	0000 1001
121	0111 0101
122	1011 0110
123	1110 0100
124	0111 1111
125	0000 0001
126	0100 0001
127	0110 1001
200	1000 1000

(a) **(i)** Show the contents of the Accumulator after execution of the instruction:

LDD 121

Accumulator:								

[1]

(ii) Show the contents of the Accumulator after execution of the instruction:

LDI 124

Accumulator:								

Explain how you arrived at your answer.

....

....

....

.... [3]

(iii) Show the contents of the Accumulator after execution of the instruction:

LDX 120

Index Register:	0	0	0	0	0	1	1	0

Accumulator:								

(b) Trace the assembly language program using the trace table.

```
300    LDD    321
301    INC
302    STO    323
303    LDI    307
304    INC
305    STO    322
306    END
307    320

320    49
321    36
322    0
323    0
```

Trace table:

Accumulator	**Memory address**			
	320	321	322	323
	49	36	0	0

[6]