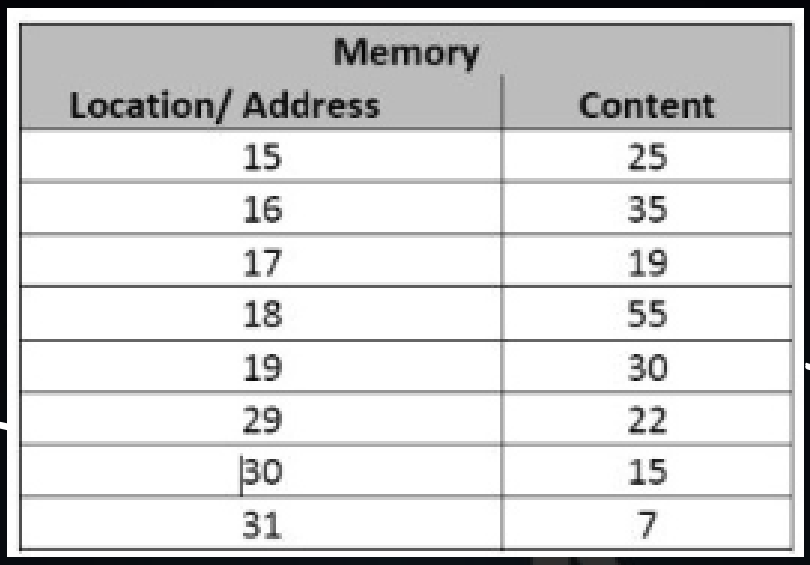
Assembly Language

Papers Dock

COMPUTER SCIENCE 9618 PAPER 1

Assembly Language

Assembly language is a low level language. It is a type of programming language which we can use to directly manipulate hardware, such as reading / writing to specific memory address. It is one step above machine code, which is the series of binary numbers (0 and 1).



Memory	
Location/ Address	Content
15	25
16	35
17	19
18	55
19	30
29	22
30	15
31	7

Value stored on location 19

Location to access the memory

Question : What are Addressing Modes

- **Methods of accessing memory are known as addressing modes**

Addressing Modes

1) Immediate Addressing

2) Direct Addressing

3) Indexed Addressing

4) Indexed Addressing

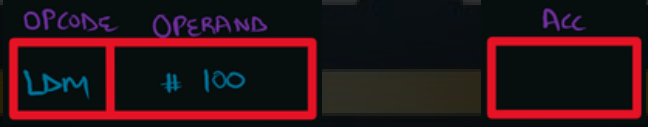
5) Relative Addressing

Concept Of Accumulator

Is a place where the results are stored (register)

Immediate Addressing

LDM : Immediate Addressing : Just load the value which is written next to the LDM instruction.



OPCODE : Instruction
OPERAND : The Data / Address on which the opcode is applied

Direct Addressing

Note : Memory will be involved

Memory	
Location/ Address	Content
15	25
16	35
17	19
18	55
19	30
29	22
30	15
31	7

Step 1 : Go to the given address
Step 2 : Load the value on that address in accumulator

Indirect Addressing

Note : Memory will be involved
(2 address)

Memory	
Location/ Address	Content
15	25
16	35
17	19
18	55
19	30
29	22
30	15
31	7

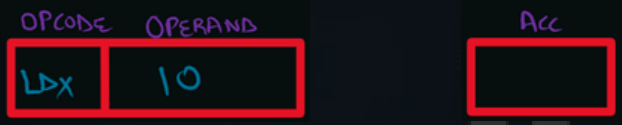
Step 1 : Go to the given address which is the address of the address
Step 2 : Go to the second address and load the value in ACC

Indexed Addressing

Note : Calculation is required
there will be register known as Index register

Memory	
Location/ Address	Content
15	25
16	35
17	19
18	55
19	30
29	22
30	15
31	7
IX	9

Step 1 : Add the initial address with IX and create a new address
Step 2 : Use the new address and load the value in ACC



Note : Relative addressing will be discussed with pastpaper example

Instruction Group

Data Movement: Accessing the value and loading in the accumulator.
Input and Output of data: Takes an input from the user and outputs the character or binary number.
Arithmetic Operations: Perform addition and subtraction.
Unconditional and conditional instructions: Move to another instruction (identified by labels).
Compare instructions: Compare the result to another value

Instruction		Explanation
Op Code	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
STO	<address>	Store the contents of ACC at the given address
ADD	<address>	Add the contents of the given address to 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)
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
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
IN		Key in a character and store its ASCII value in ACC
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system

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									

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:								

(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]

(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]

Relative And Symbolic Addressing

(b) 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	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.

Complete the trace table for the following assembly language program.

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		

(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

Symbol table

Symbolic address	Absolute address

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)

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

- The assembler scans the assembly language instruction in sequence.
- When it meets a symbolic address, it checks to see if it is already in the symbol table.
- If not, it adds it to the symbol table in the symbolic address column.
- If it is already in the symbol table, check if the absolute address is known.
- If the absolute address is known, it is entered in the appropriate cell.
- If the absolute address is not known, then leave "Unknown."

Question : The assembler must then complete the second pass building up the executable file.

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

- **Instruction Table**

Two Pass Assembler

The assembly language program is converted into machine code by a two-pass assembler.

First Pass

- **Reads the source code one line at a time.**
- **Removes white space.**
- **Adds label to the symbol table.**
- **Expansion of macros.**
- **Deals with directives.**
- **Checks the opcode is in the instruction set.**
- **Removes comments.**

Second Pass

- **Reads the source code one line at a time.**
- **generates object code**

Macros: Set of instructions that perform a specific task, also called subroutines, can be called many times during the execution of code.

Directive: Is a special instruction that guides the assembler on how to handle the assembly process.

Assembly Language

Question 1

(b) 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
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
MOV	<register>	Move the contents of the accumulator to the given register (IX)
STO	<address>	Store contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the 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
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
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 address or a symbolic address # denotes a denary number, e.g. #123		

The current contents of the main memory and selected values from the ASCII character set are shown.

Address	Instruction
200	LDD 365
201	CMP 366
202	JPE 209
203	INC ACC
204	STO 365
205	MOV IX
206	LDX 365
207	OUT
208	JMP 200
209	END
...	
365	1
366	3
367	65
368	66
IX	0

ASCII code table (selected codes only)

ASCII code	Character
65	A
66	B
67	C
68	D

Complete the trace table for the program currently in main memory.

Instruction address	ACC	Memory address				IX	Output
		365	366	367	368		
		1	3	65	66	0	

[6]

Question 2

4 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
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
STO	<address>	Store contents of ACC at the given address
ADD	<address>	Add the contents of the given address to 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)
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
JMP	<address>	Jump to the given address
IN		Key in a character and store its ASCII value in ACC
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system
# denotes a denary number, e.g. #123		

The current contents of the main memory and selected values from the ASCII character set are:

Address	Instruction
70	IN
71	CMP 100
72	JPE 80
73	CMP 101
74	JPE 76
75	JMP 80
76	LDD 102
77	INC ACC
78	STO 102
79	JMP 70
80	LDD 102
81	DEC ACC
82	STO 102
83	JMP 70
...	
100	68
101	65
102	100

ASCII code table (selected codes only)

ASCII code	Character
65	A
66	B
67	C
68	D

(a) Complete the trace table for the program currently in main memory when the following characters are input:

```
A D
```

Do not trace the program any further when the third input is required.

Instruction address	ACC	Memory address		
		100	101	102
		68	65	100

[4]

Question 3

(b) 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
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
STO	<address>	Store the contents of ACC at the given address
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>
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
JMP	<address>	Jump to the given address
IN		Key in a character and store its ASCII value in ACC
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system
XOR	#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	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>
AND	#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>
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. #123		

The current contents of main memory are shown:

Address	Data
100	01010101
101	11110000
102	00001111
103	00000000
104	11111111

(ii) The following table contains five assembly language instruction groups.

Write an appropriate assembly language instruction for each instruction group, using the given instruction set. The first one has been completed for you.

Instruction Group	Instruction
Data movement	`LDM #2`
Input and output of data	
Arithmetic operations	
Unconditional and conditional instructions	
Compare instructions	

[4]

(iii) The opcode `LDM` uses immediate addressing. The opcode `LDD` uses direct addressing.

Identify **and** describe **one additional** mode of addressing.

Mode of addressing .....

Description .....

.....

.....

[2]

Question 4

(c) Tick (✓) **one or more** boxes in each row to indicate whether the task is performed in the first pass or the second pass of a two-pass assembler.

Task	First pass	Second pass
Remove comments.		
Read the assembly language program one line at a time.		
Generate the object code.		
Check the opcode is in the instruction set.		

[2]

Question 5

7 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
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
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 the contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the ACC
ADD	#n	Add the denary number n to the ACC
INC	<register>	Add 1 to 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>
CMI	<address>	Indirect addressing. The address to be used is at the given address. Compare the contents of ACC with the contents of this second 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
END		Return control to the operating system
<address> can be an absolute or symbolic address # denotes a denary number, e.g. #123 B denotes a binary number, e.g. B01001101		

(a) Trace the program currently in memory using the trace table, stopping when line `90` is executed for a second time.

Address	Instruction
75	`LDR #0`
76	`LDX 110`
77	`CMI 102`
78	`JPE 91`
79	`CMP 103`
80	`JPN 84`
81	`ADD 101`
82	`STO 101`
83	`JMP 86`
84	`INC ACC`
85	`STO 101`
86	`LDD 100`
87	`INC ACC`
88	`STO 100`
89	`INC IX`
90	`JMP 76`
91	`END`
...	
100	`0`
101	`0`
102	`112`
103	`4`
...	
110	`1`
111	`4`
112	`0`

Instruction address	ACC	IX	Memory address						
			100	101	102	103	110	111	112
			0	0	112	4	1	4	0

[5]

Question 6

6 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
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
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 the ACC
ADD	#n	Add the denary number n to the ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
CMP	#n	Compare the contents of ACC with 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
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system
<address> can be an absolute or a symbolic address # denotes a denary number, e.g. #123 B denotes a binary number, e.g. B01001101		

(a) The current contents of main memory and selected values from the ASCII character set are given.

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

Address	Instruction
77	`LDR #0`
78	`LDX 110`
79	`CMP #35`
80	`JPE 92`
81	`ADD 100`
82	`STO 101`
83	`LDM #1`
84	`ADD 100`
85	`STO 100`
86	`INC IX`
87	`LDX 110`
88	`CMP #35`
89	`JPN 81`
90	`LDD 100`
91	`ADD #48`
92	`OUT`
93	`END`
…	
100	0
101	0
…	
110	66
111	65
112	35

ASCII value	Character
49	1
50	2
51	3
52	4
65	A
66	B
67	C
68	D

Instruction address	ACC	IX	Memory address					Output
			100	101	110	111	112	
			0	0	66	65	35	

[4]

(ii) The following instructions are repeated for your reference.

Instruction		Explanation
Opcode	**Operand**	
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC
STO	<address>	Store contents of ACC at the given address

State the purpose of this part of an assembly language program.

```
LDD 100
STO 165
LDD 101
STO 100
LDD 165
STO 101
```

.....

.....

[1]

Question 7

6 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
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
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
MOV	<register>	Moves the contents of the accumulator to the given register (IX)
STO	<address>	Store contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the ACC
ADD	#n	Add the denary number n to the ACC
SUB	#n	Subtract the denary number n from the ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
JMP	<address>	Jump to the given address
CMP	#n	Compare the contents of ACC with number n
CMI	<address>	Indirect addressing. The address to be used is at the given address. Compare the contents of ACC with the contents of this second 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
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system
<address> can be an absolute or a symbolic address # denotes a denary number, e.g. #123 B denotes a binary number, e.g. B01001101		

(a) The current contents of main memory and selected values from the ASCII character set are given.

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

Address	Instruction
75	LDR #0
76	LDD 100
77	CMP #2
78	JPE 91
79	LDX 110
80	SUB #32
81	CMP #65
82	JPN 86
83	LDM #1
84	ADD 101
85	STO 101
86	LDM #1
87	ADD 100
88	STO 100
89	INC IX
90	JMP 76
91	LDD 101
92	ADD #48
93	OUT
94	END
...	
100	1
101	0
...	
110	97
111	98
112	97

ASCII value	Character
49	1
50	2
51	3
52	4
65	A
66	B
67	C
68	D

Instruction address	ACC	IX	Memory address					Output
			100	101	110	111	112	
			1	0	97	98	97	

[4]

(ii) Explain the purpose of **relative addressing** in an assembly language program.

.....

.....

.....

..... [2]

(c) One instruction group is data movement.

Give the name of **one other** instruction group.

.....

..... [1]

Question 8

3 A program is written in assembly language.

(a) The program is converted into machine code by a two-pass assembler.

Draw **one or more** lines to identify the pass or passes in which each action takes place.

Action	Pass
generates object code	
reads the source code one line at a time	first
removes white space	second
adds labels to the symbol table	

[3]

(b) Assembly language statements can use different modes of addressing.

Complete the following description of addressing modes.

..... addressing is when the operand holds the memory address of the data.

..... addressing is when the operand holds a memory address that stores the memory address of the data.

..... addressing is when the operand is the data.

[3]

Question 9

8 **(a)** Identify the purpose of the first pass of a two-pass assembler.

.....

..... [1]

(b) 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**	
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 the ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
CMP	#n	Compare the contents of ACC with number n
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
OUT		Output to the screen the character whose ASCII value is stored in ACC
<address> can be an absolute or symbolic address # denotes a denary number, e.g. #123		

(i) Give **one** example of an instruction that belongs to **each** of the following instruction groups.

Only use the instructions given in the table. Each instruction must have a suitable operand.

Data movement .....

Arithmetic operation .....

Conditional instruction .....

[3]

(ii) The instruction `LDR #2` uses immediate addressing.

Give **one** similarity and **one** difference between direct addressing and indexed addressing.

Similarity .....

.....

.....

Difference .....

.....

.....

[2]

(iii) Identify **one other** mode of addressing.

.....

..... [1]

Question 10

9 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**	
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 the contents of ACC at the given address
ADD	#n	Add the denary number n to the ACC
JMP	<address>	Jump to the given address
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>
CMI	<address>	Indirect addressing. The address to be used is at the given address. Compare the contents of ACC with the contents of this second address
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
IN		Key in a character and store its ASCII value in ACC
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system
<address> can be an absolute or a symbolic address # denotes a denary number, e.g. #123		

(a) The instructions in the processor's instruction set can be grouped according to their function.

Identify the instruction group for each of the following opcodes.

`IN` .....

`ADD` .....

`JPE` .....

`CMI` .....

[4]

(b) The current contents of main memory and selected values from the ASCII character set are given on page 15.

Trace the program currently in memory using the trace table when the input is '1'.

Address	Instruction
10	LDR #0
11	IN
12	STO 101
13	LDX 110
14	CMP 100
15	JPE 21
16	LDD 101
17	ADD #16
18	INC IX
19	STO 100
20	JMP 13
21	OUT
22	END
...	
100	0
101	0
...	
110	51
111	65

ASCII value	Character
49	1
50	2
51	3
52	4
...	
65	A
66	B
67	C
68	D

Instruction address	ACC	IX	Memory address				Output
			100	101	110	111	
			0	0	51	65	

[4]

Question 11

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]

Question 12

3 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
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX)
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
48	51
49	6
50	48
51	50
52	49
53	50
54	6

The current contents of the ACC and IX are shown:

ACC	2
IX	50

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

Program number	Code	ACC content
1	LDM #50 INC ACC SUB #1	
2	LDI 51 ADD 52	
3	LDR #2 LDX 50 DEC ACC	
4	LDD 52 SUB 54 INC ACC	

[4]

Question 13

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 14

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.

.....

.....

.....

.....

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 15

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 16

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 17

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]

Question 18

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 19

(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]

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 20

(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]

(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 21

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

Relative addressing .....

.....

.....

Indexed addressing .....

.....

..... [2]

6 16

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 22

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

Main Memory

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

(i) LDM #500

ACC .....[1]

(ii) LDD 500

ACC .....[1]

(iii) LDX 500

ACC .....[1]

(iv) LDI 500

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 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 23

Instruction			Explanation
Op code (mnemonic)	Operand	Op code (binary)	
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

(i) LDD 355

ACC ..... [1]

(ii) LDM #355

ACC ..... [1]

(iii) LDX 351

ACC ..... [1]

(iv) LDI 355

ACC ..... [1]

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

(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 24

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 25

(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	{	|	}

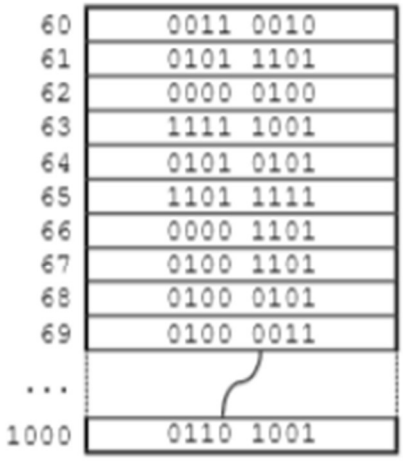
Trace table:

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 26

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



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 27

The diagram shows the contents of the index register:

Index register:

1	1	0	0	1	1	0	1

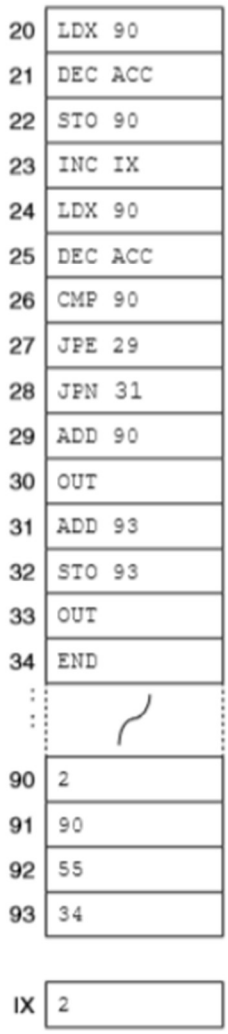
(a) Show the contents of the index register after the execution of the instruction:

```
INC IX
```

Index register:

[1]

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



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 28

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]

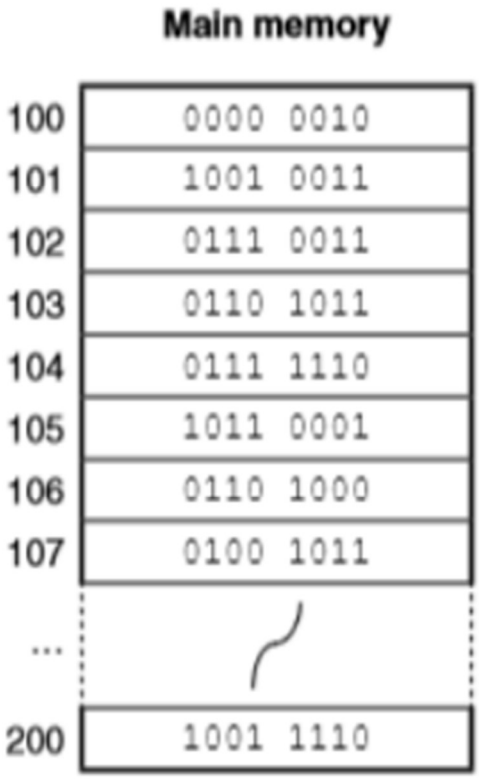
(ii) There is a redundant instruction in the code in part (b)(i).

State the address of this instruction.

.....[1]

Question 29

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



(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.

.....

.....

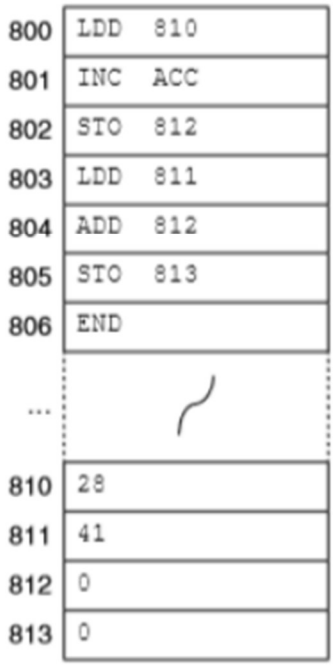
.....

.....

.....

..... [3]

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



Trace table:

ACC	Memory address			
	810	811	812	813
	28	41	0	0

[6]

Question 30

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 31

The diagram shows the contents of the memory.

Main memory

120	0 0 0 0 1 0 0 1
121	0 1 1 1 0 1 0 1
122	1 0 1 1 0 1 1 0
123	1 1 1 0 0 1 0 0
124	0 1 1 1 1 1 1 1
125	0 0 0 0 0 0 0 1
126	0 1 0 0 0 0 0 1
127	0 1 1 0 1 0 0 1
200	1 0 0 0 1 0 0 0

(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]

Answer

Answer 1

3(b) — **1 mark** for each shaded set of values — **6**

Instruction address	ACC	Memory address				IX	Output
		365	366	367	368		
		1	3	65	66	0	
200	1						
201							
202							
203	2						
204		2					
205						2	
206	65						
207							A
208							
200	2						
201							
202							
203	3						
204		3					
205						3	
206	66						
207							B
208							
200	3						
201							
202							
209							

Answer 2

4(a)	**1 mark** for each shaded section / bullet point • Load 65 into ACC • Load 100 into ACC, increment and store in 102 • Load 68 into ACC • Load 101 into ACC, decrement and store in 102	**4**

Instruction address	ACC	Memory address		
		100	101	102
		68	65	100
70	65			
71				
72 73				
74				
76 77 78	100 101			 101
79				
70 71	68			
72				
80 81	101 100			
82				100
83				
(70)				

Answer 3

8(b)(ii)	**1 mark** for each correct instruction **Instruction Group** / **Instruction** Data movement — `LDM #2` Input and output of data — `IN / OUT` Arithmetic Operations — `INC ACC / INC IX` Unconditional and conditional instructions — `JPN 100 / JMP 100` Compare instructions — `CMP 100`	**4**
8(b)(iii)	**1 mark** for name, **1 mark** for description • Indirect addressing • the address to be used is at the given address • Relative addressing • the address to be used is an offset number of locations away, relative to the address of the current instruction • Indexed addressing • form the address from the given address plus the contents of the index register	**2**

Answer 4

6(c)	**1 mark** for each pair of highlighted rows **Task** / **First pass** / **Second pass** Remove comments. — ✓ Read the assembly language program one line at a time. — ✓ — ✓ Generate the object code. — ✓ Check the opcode is in the instruction set. — ✓	**2**

Answer 5

7(a)	**1 mark** for each set of highlighted rows.	**5**

Instruction address	ACC	IX	Memory address						
			100	101	102	103	110	111	112
			0	0	112	4	1	4	0
75		0							
76	1								
77									
78									
79									
80									
84	2								
85				2					
86	0								
87	1								
88			1						
89		1							
90									
76	4								
77									
78									
79									
80									
81	6								
82				6					
83									
86	1								
87	2								
88			2						
89		2							
90									

Answer 6

6(a)(i)	**1 mark** for each set of highlighted rows.	**4**

Instruction address	ACC	IX	Memory address					Output
			100	101	110	111	112	
			0	0	66	65	35	
77		0						
78	66							
79								
80								
81								
82				66				
83	1							
84								
85			1					
86		1						
87	65							
88								
89								
81	66							
82								
83	1							
84	2							
85			2					
86		2						
87	35							
88								
89								
90	2							
91	50							
92								2
93								

6(a)(ii)	swaps the contents of memory address 100 and 101	**1**

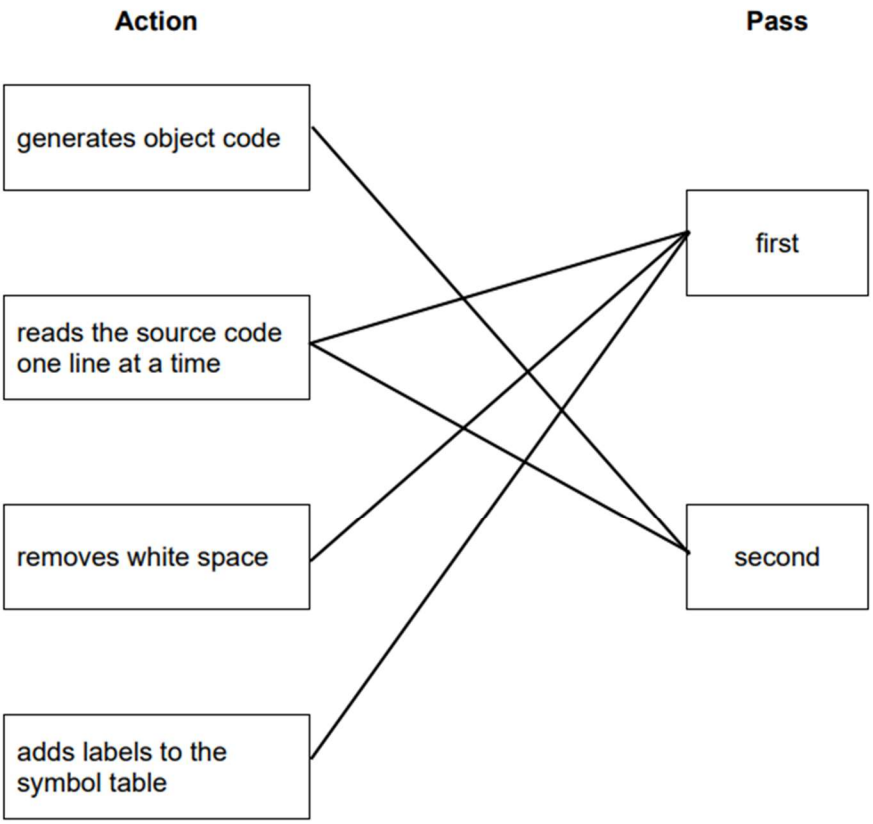
Answer 7

6(a)(i)	**1 mark** for each set of highlighted rows:	**4**

Instruction address	ACC	IX	Memory address					Output
			100	101	110	111	112	
			1	0	97	98	97	
75		0						
76	1							
77								
78								
79	97							
80	65							
81								
82								
83	1							
84								
85				1				
86								
87	2							
88			2					
89		1						
90								
76								
77								
78								
91	1							
92	49							
93								1
94								

6(a)(ii)	**1 mark** for each bullet point: • To allow for **re-locatable** code • ... because all (target) addresses can be specified by the base address + offset	**2**
6(c)	**1 mark** for a correct name: • input and output of data • arithmetic operations • unconditional and conditional instructions • compare instructions	**1**

Answer 8

3(a)	**1 mark** for generates object code to second pass **1 mark** for reads source code one line at a time to both boxes **1 mark** for removes white space and adds labels to first pass 	**3**

3(b)	**1 mark** for each correct term **Direct** addressing is when the operand holds the memory address of the data. **Indirect** addressing is when the operand holds a memory address that stores the memory address of the data. **Immediate** addressing is when the operand is the data.	**3**

Answer 9

8(a)	**1 mark** for: • To **create** a symbol table	**1**
8(b)(i)	**1 mark** for each bullet point • Data movement: e.g. LDR #50 // STO 201 • Arithmetic operation: e.g. ADD 100 // INC IX • Conditional instruction: e.g. JPE 96	**3**
8(b)(ii)	**1 mark** for each bullet point (**max 2**) Similarity: • both load the **contents** of an address into the Accumulator Difference: • direct accesses the address given by the operand whereas indexed adds the contents of IX to the operand and accesses the data at that calculated address	**2**
8(b)(iii)	**1 mark** for • Indirect (addressing) • Relative (addressing)	**1**

Answer 10

9(a)	**1 mark** for each bullet point • `IN` - Input **and** output of data • `ADD` - Arithmetic operations • `JPE` - Unconditional and conditional instructions • `CMI` - Compare instructions	**4**
9(b)	**1 mark** for each set of shaded rows	**4**

Instruction address	ACC	IX	Memory address				Output
			100	101	110	111	
			0	0	51	65	
10		0					
11	49						
12				49			
13	51						
14							
15							
16	49						
17	65						
18		1					
19			65				
20							
13	65						
14							
15							
21							A
22							

Answer 11

4(a)	**1 mark** for each correct answer:	**4**

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**

Answer 12

3(a)	**1 mark** for each correct answer:	**4**

Program Number	Code	ACC Content
1	LDM #50 INC ACC SUB #1	**50**
2	LDI 51 ADD 52	**97**
3	LDR #2 LDX 50 DEC ACC	**48**
4	LDD 52 SUB 54 INC ACC	**44**

Answer 13

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

Answer 14

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

5(c) **1 mark** for each highlighted section block

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					

Answer 15

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										

Answer 16

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)</td><td>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)</td><td>2</td></tr>
<tr><td>4(b)</td><td>1 mark for JPN correct
1 mark for both of ADD and DEC correct
1 mark for LDR correct

<table>
<tr><th></th><th>Description</th><th>Jump instruction</th><th>Arithmetic operation</th><th>Data movement</th></tr>
<tr><td>LDR #3</td><td>Load the number 3 to the Index Register</td><td></td><td></td><td>✓</td></tr>
<tr><td>ADD #2</td><td>Add 2 to the Accumulator</td><td></td><td>✓</td><td></td></tr>
<tr><td>JPN 22</td><td>Move to the instruction at address 22</td><td>✓</td><td></td><td></td></tr>
<tr><td>DEC ACC</td><td>Subtract 1 from the Accumulator</td><td></td><td>✓</td><td></td></tr>
</table>
</td><td>3</td></tr>
</table>

Answer 17

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										
52	5									
53										
54										
55	69									
56									E	[1]
57	1									
58	2									
59							2			
60								2		
61										
52	0									
53										[1]
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

Answer 18

Question	Answer	Marks
7	**1 mark** for each correct addressing mode	5

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

Answer 19

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)(ii)</td><td>1 mark per bullet point

Direct addressing:
□ 20 is the address of the data

Indirect addressing:
□ 20 is an address which holds the address where the data is stored</td><td>2</td></tr>
<tr><td>4(b)</td><td>1 mark for 1 correct tick
2 marks for 3 correct ticks

<table><tr><th>Instruction</th><th>Symbolic</th><th>Absolute</th></tr><tr><td>ADD 90</td><td></td><td>✓</td></tr><tr><td>CMP found</td><td>✓</td><td></td></tr><tr><td>STO 20</td><td></td><td>✓</td></tr></table></td><td>2</td></tr>
<tr><td>4(c)(i)</td><td>186</td><td>1</td></tr>
<tr><td>4(c)(ii)</td><td>BA</td><td>1</td></tr>
<tr><td>4(c)(iii)</td><td>–70</td><td>1</td></tr>
</table>

Instruction address	ACC	Memory address						IX (Index Register)	OUTPUT
		200	201	202	203	204	205		
70	130	130	133	130	0	0	2	0	
71									*
72					130				
73	0								
74	1								
75						1			
76								1	
77	133								
78									
79									
81	1								
82									
83									
74	2								
75						2			
76								2	
77	130								
78									
79									
80									*
81	2								
82									
83									
84									

Answer 20

3(a)(i)	**1 mark** per bullet point Absolute addressing: • The operand is a numeric address // The numeric address is given // referring directly to a memory location Symbolic addressing: • The operand is a word/symbol // A word/symbol represents the memory location/address	**2**
3(a)(ii)	**1 mark** per example Absolute addressing: For example, `ADD 230` Symbolic addressing: For example, `ADD num1`	**2**
3(b)(i)	**1 mark** per bullet point Indexed addressing: • The address to be used is formed by: operand + the contents of the Index Register (IX) Immediate addressing: • The operand is not an address // the operand is the actual value to be loaded	**2**

Question	Answer	Marks
3(b)(ii)	**1 mark** per example Indexed: For example, `LDX 20` Immediate: For example, `ADD #20`	**2**
3(c)(i)	193	**1**
3(c)(ii)	C1	**1**
3(c)(iii)	–63	**1**

Question	Answer	Marks

Instruction address	ACC	Memory address						IX	OUTPUT
		100	101	102	103	104	105		
		2	302	303	303	0	303	1	
40	2								
41									
43	302								
44									
45									
47	2								
48	1								
49		1							
50								2	
51									
41									
42									
43	303								
44									
45									
46									+
47	1								
48	0								
49		0							
50								3	
51									
41									
54									

Answer 21

Question	Answer	Marks
2(b)(ii)	**F2**	1
2(b)(iii)	**–14**	1
2(b)(iv)	1 1 1 1 0 0 1 1	1
2(c)	**1 mark** per bullet point • Loading 8 (instruction 20) • Comparison and loading 453 (instructions 21–23) • Outputting & (instruction 26) • Loading, incrementing and storing in 96 (instructions 27–29) • Incrementing Index Register (instruction 30) • Jumping and loading 452 (instruction 23) • Jumping, loading, incrementing, storing in 96, incrementing IX and end (instructions 24–32)	7

Instruction address	ACC	Memory address						IX	OUTPUT
		93	94	95	96	97	98		
		453	453	452	8	10	453	8	
20	8								
21									
22									
23	453								
24									
25									
26									&
27	8								
28	9								
29					9				
30								9	
31									
21									
22									
23	452								
24									
25									
27	9								
28	10								
29					10				
30								10	
31									
21									
22									
32									

Answer 22

Question	Answer	Marks
4(a)(i)	500	1
4(a)(ii)	496	1
4(a)(iii)	502	1
4(a)(iv)	86	1
4(b)	0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 Both correct op codes 1 Operand 0001 0001 1 Operand 0110 0001 1	3
4(c)	256	1
4(d)(i)	07 C2 07 1 C2 1	2
4(d)(ii)	LDI 63 LDI 1 63 1	2

Answer 23

Question	Answer	Marks
5(a)(i)	351	1
5(a)(ii)	355	1
5(a)(iii)	22	1

Question	Answer	Marks
5(a)(iv)	86	1
5(b)	Op code Operand 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 1 Both correct op codes 1 Operand 0100 0011 1 Operand 0000 0111 1	3
5(c)(i)	14 5E 14 1 5E 1	2
5(c)(ii)	LDR #77 LDR 1 #77 1	2

Answer 24

<table>
<tr>
<td>4(b)</td>
<td>
<table>
<tr><th>ACC</th><th>CountDown</th><th>OUTPUT</th><th></th></tr>
<tr><td></td><td>15</td><td></td><td></td></tr>
<tr><td>67</td><td></td><td>C</td><td></td></tr>
<tr><td>15</td><td></td><td></td><td></td></tr>
<tr><td>14</td><td>14</td><td></td><td>(1)</td></tr>
<tr><td>51</td><td></td><td>3</td><td>(1) + (1)</td></tr>
<tr><td>14</td><td></td><td></td><td>(1)</td></tr>
<tr><td>13</td><td>13</td><td></td><td rowspan="3">(1)</td></tr>
<tr><td>32</td><td></td><td></td></tr>
<tr><td>88</td><td></td><td>X</td></tr>
</table>
</td>
<td>5</td>
</tr>
<tr>
<td>4(c)</td>
<td>Three marks from:

▫ The assembler scans the assembly language instructions in sequence
▫ When it meets a symbolic address checks to see if already in symbol table
▫ If not, it adds it to the symbol table in the symbolic address column
▫ If it is already in symbol table check if absolute address known
▫ If the absolute address is known, it is entered in the appropriate cell
▫ If the absolute address is not known mark / leave as unknown</td>
<td>Max 3</td>
</tr>
<tr>
<td>4(d)(i)</td>
<td>The op code / mnemonic / instruction table</td>
<td>1</td>
</tr>
<tr>
<td>4(d)(ii)</td>
<td>A – 1110 0110 0110 1000
(1) (1)

B – E6 68 (1)</td>
<td>3</td>
</tr>
</table>

Answer 25

(b)

Instruction address	Working space	ACC	Memory address				IX	OUTPUT
			100	101	102	103		
			20	100	1	0	1	
50		20						
51		21						
52						21		
53		100						
54		120						
55								
56								
57								
59							2	
60		20						
61		120						
62								'x'
63								

Answer 26

9 (a) (i) **One mark** for the contents of the accumulator and **one mark** for the reason. **[2]**

Accumulator contents: 0100 0101

Reason:

Address is 60
Contents of the index register is 8
And 60 + 8 = 68 in denary gives the address
The contents of which is 0100 0101 in binary.

(ii) 0000 0111 **[1]**

Answer 27

4 (a) 11001110 [1]

(b) [7]

Instruction	Working space	ACC	Memory address				IX	OUTPUT
			90	91	92	93		
			2	90	55	34	2	
20		55						
21		54						
22			54					
23							3	
24		34						
25		33						
26								
27								
28								
31		67						
32						67		
33								'C'
34								

Answer 28

8 (a) (i)

Accumulator:	1	0	0	1	0	1	1	1

[1]

(ii) **One mark** for answer and **two marks** for explanation

Accumulator:	1	1	0	0	0	0	1	0

- Index Register contains 1001 = 9
- 800 + 9 = 809

[3]

(b) (i) **ONE** mark for each correct row.

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

[4]

(ii) **107** [1]

Answer 29

(a) (i)

0	1	1	1	0	0	1	1

[1]

(ii) **ONE** mark for Accumulator contents, **ONE** mark for the explanation.

1	0	1	1	0	0	0	1

- Index Register holds the value 4; 101 + 4 = 105 so load data from address 105

[2]

(iii) **ONE** mark for Accumulator contents, **TWO** marks for the explanation.

0	1	0	0	1	0	1	1

- Memory address 103 contains the value 107
- So address 107 is the address from which to load the data

[3]

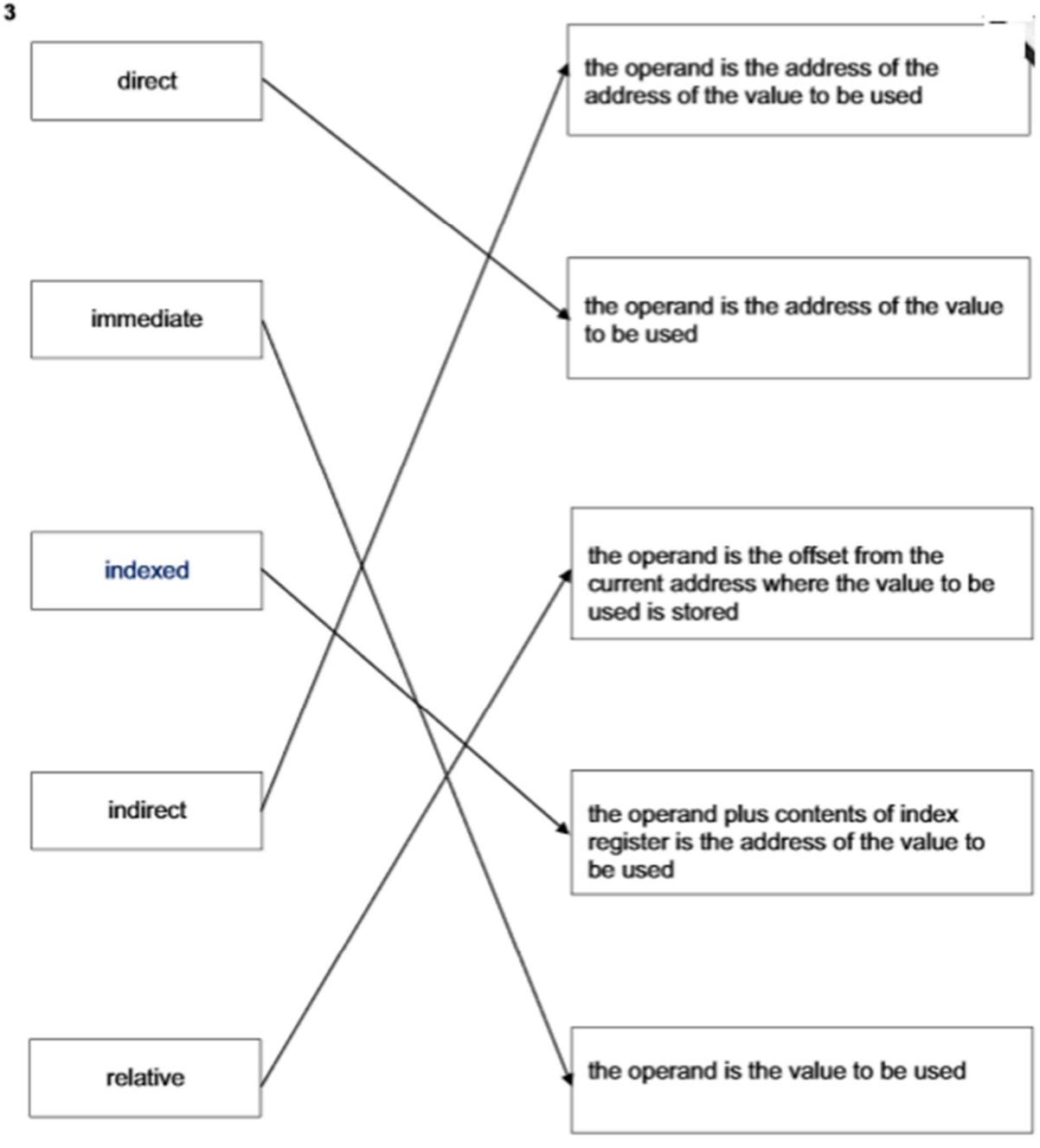
(b) **ONE** mark for each correct row.

ACC	Memory address			
	810	811	812	813
	28	41	0	0
28				
29				
			29	
41				
70				
				70

[6]

Answer 30

3


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

[4]

Answer 31

(a) (i)

Accumulator:	0	1	1	1	0	1	0	1

[1]

(ii)

Accumulator:	0	1	1	0	1	0	0	1

[1]

explanation

- content of 124 is 0 1 1 1 1 1 1 1
- this is equivalent to 127
- contents of 127 are 0 1 1 0 1 0 0 1 [2]

(iii)

Accumulator:	0	1	0	0	0	0	0	1

[1]

explanation

- index register value = 6
- 120 + 6 = 126
- contents of 126 placed in the accumulator [2]

(b) 1 mark for each correct value in the table.

Accumulator	Memory address			
	320	321	322	323
	49	36	0	0
36				
37				
				37
49				
50				
			50	

[6]