QUESTION 3.

7 The table shows assembly language instructions for a processor which has one register, the Accumulator (ACC).

Instruction		Explanation
Op code	**Operand**	
LDD	<address>	Direct addressing. Load contents of given address to ACC
STO	<address>	Store the contents of ACC at the given address
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
INC	<register>	Add 1 to contents of the register (ACC)
JMP	<address>	Jump to the given address
END		Return control to operating system

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:								

Explain how you arrived at your answer.

....

....

....

.... [3]

QUESTION 4.

9 The table shows assembly language instructions for a processor which has one register, the Accumulator (ACC) and an index register (IX).

Instruction		Explanation
Op code	**Operand**	
`LDD`	`<address>`	Direct addressing. Load the contents of 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.
`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 screen the character whose ASCII value is stored in ACC.
`END`		Return control to the operating system.

(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 instructi

```
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]

(b) Complete the trace table on the opposite page for the following assembly lan

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 5.

4 The table shows assembly language instructions for a processor which has one register, the Accumulator (ACC) and an index register (IX).

Instruction		Explanation
Op code	Operand	
LDD	<address>	Direct addressing. Load the contents of 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.
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 screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

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 lan

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 6.

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

Instruction		**Explanation**
Op code	**Operand**	
`LDD`	`<address>`	Direct addressing. Load the contents of 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.
`STO`	`<address>`	Store contents of ACC at the given address.
`ADD`	`<address>`	Add the contents of the given address to ACC.
`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.

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 languag
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.

....

(c) The program used the ASCII coding system for character codes. An alternative coding system is Unicode.

(i) Give **two** disadvantages of using ASCII code.

1 ....

....

2 ....

....[2]

(ii) Describe how Unicode is designed to overcome the disadvantages of ASCII.

....

....

....

....[2]

BLANK PAGE

BLANK PAGE

QUESTION 7.

7 One management task carried out by an operating system is to provide a user interface.

Describe **two** more of these management tasks.

1 ....

....

....

....

2 ....

....

....

....[4]

QUESTION 8.

6 **(a)** The operating system (OS) contains code for performing various manageme

The appropriate code is run when the user performs various actions.

Draw a line to link each OS management task to the appropriate user action.

OS management task	Action
Main memory management	The user moves the mouse on the desktop
Input/Output management	The user closes the spreadsheet program
Secondary storage management	The user selects the Save command to save their spreadsheet file
Human computer interface management	The user selects the Print command to output their spreadsheet document

[3]

(b) A user has the following issues with the use of his PC.

State the utility software which should provide a solution.

(i) The hard disk stores a large number of video files. The computer frequently runs out of storage space.

Utility software solution ....[1]

(ii) The user is unable to find an important document. He thinks it was deleted in error some weeks ago. This must not happen again.

Utility software solution ....[1]

(iii) The operating system reports 'Bad sector' errors.

Utility software solution ....[1]

(iv) There have been some unexplained images and advertisements appearing on the screen. The user suspects it is malware.

Utility software solution ....[1]

QUESTION 9.

4 **(a)** **(i)** Explain why a personal computer (PC) needs an operating system (OS)

....

....

....[2]

(ii) One of the tasks carried out by the OS is the management of the use of the processor.

Name and describe **two** other management tasks that the OS performs.

1 ....

....

....

2 ....

....

....

[4]

(b) A user has the following issues with the use of their personal computer (PC).

For each case, state the utility software which should provide a solution.

(i) The user wants to send a large file as an attachment to an email. The user knows that the recipient's Internet Service Provider (ISP) has a limit of 2MB for file attachments.

Utility software solution: ....[1]

(ii) The user is writing a book and is worried that the document files could get damaged or deleted.

Utility software solution: ....[1]

(iii) The computer has recently been slow to load large files. The user has deleted a large number of small files to try to solve the problem. A friend has advised that there is a procedure which should be regularly carried out to reorganise file storage on the hard disk.

Utility software solution: ....[1]

(iv) The user clicked on an attachment in an unsolicited email. Since then, the computer has shown some unexplained behaviours.

Utility software solution: ....[1]

QUESTION 10.

1 One of the tasks of the operating system (OS) is the management of the mai
computer system.

State and describe **three** other operating system management tasks.

1 ....

....

....

2 ....

....

....

3 ....

....

....

[6]

QUESTION 11.

4 The table shows assembly language instructions for a processor which has one register, the Accumulator (ACC) and an index register (IX).

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.

(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 symbo absolute addressing.

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

(d) The current contents of the main memory, Index Register (IX) and selected ASCII character set are provided with a copy of the instruction set.

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.

Complete the trace table for the given assembly language program.

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

[8]

QUESTION 12.

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

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 compare instruction, jump to <address> if the compare was True.
`JPN`	`<address>`	Following 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.

(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 addressi**

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 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	=

This is a copy of the 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.

Complete the trace table for the given assembly language program.

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

[7]

QUESTION 13.

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

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 compare instruction, jump to <address> if the compare was True.
`JPN`	`<address>`	Following 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.

(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]

(c) The current contents of the main memory, Index Register (IX) and selected ASCII character set are provided with a copy of the instruction set.

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>.
`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.

Complete the trace table for the given assembly language program.

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

[7]

QUESTION 14.

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

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

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

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

[2]

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

Identify **two** other groups of instructions.

1 ....

....

2 ....

....

[2]

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

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

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

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

ASCII code table (Selected codes only)

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

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

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

[8]

QUESTION 15.

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

```
01      MAR ← [PC]

02      PC ← [PC] - 1

03      MDR ← [MAR]

04      CIR ← [MAR]
```

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

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

Line number ....

Correction ....

Line number ....

Correction ....

Line number ....

Correction ....

[3]

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

Identify **two** other groups of instructions.

1 ....

....

2 ....

....

[2]

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

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

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

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

IX	0

ASCII code table (Selected codes only)

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

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

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

[8]

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

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

[1]

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

.... [1]

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

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

.... [1]

QUESTION 16.

1 A computer has an operating system (OS) and utility software.

(a) The following table lists key management tasks performed by an operating s their descriptions.

Complete the table by writing the missing management task names and descriptions.

Management task	**Description**
Memory management	
	Provides user accounts and passwords
	Handles the signals sent when the attention of the processor is required elsewhere
Provision of a software platform	

[4]

(b) A hard disk formatter and a hard disk defragmenter are two examples of utility software.

(i) Describe the actions performed by a hard disk formatter and a hard disk defragmenter.

Hard disk formatter ....

....

....

....

Hard disk defragmenter ....

....

....

....

[4]

(ii) Identify **three other** examples of utility software that can be installed on

1 ....

....

2 ....

....

3 ....

....

[3]

QUESTION 17.

2 Aaron uses a desktop computer to do school work.

(a) Aaron has a mouse and keyboard that he can use as input devices and a mo. output device.

(i) Identify **two** additional input devices Aaron could use with his desktop computer.

1 ....

2 ....
[2]

(ii) Identify **two** additional output devices Aaron could use with his desktop computer.

1 ....

2 ....
[2]

(iii) Aaron needs to store a large number of applications and data on his computer. He needs at least 50GB of secondary storage space.

Identify **one** internal secondary storage device for Aaron's computer.

....

.... [1]

(iv) Describe the internal operation of a trackerball mouse.

....

....

....

....

....

.... [3]

(b) Aaron’s computer has an operating system (OS). The OS manages the ru and provides a user interface.

Describe these OS management tasks.

Process management ....

....

....

....

....

....

Provision of a user interface ....

....

....

....

....

....

[6]

(c) Aaron’s computer has a virus checker and backup software.

Describe these utility programs.

Virus checker ....

....

....

....

Backup software ....

....

....

....

[4]

(d) Aaron creates a web page using JavaScript code and HTML tags.

Describe how the JavaScript code is translated using an interpreter.

....

....

....

.... [2]

QUESTION 18.

1 **(a)** The diagram shows different types of software on the left, and descriptions o

Draw a line from each type of software to its correct description.

Type of software

Operating system

Utility program

Library program

Compiler

Description

Provides a ready-built routine that can be imported into a program

Provides an interface between the user and the hardware

Converts source code into a low-level language

Creates a new document for the user to edit

An additional program that helps to maintain or configure the system

[4]

(b) Describe the purpose of disk repair software.

....

....

....

....

....

.... [3]

QUESTION 19.

2 Leonardo’s mobile phone has an operating system (OS).

(a) Describe the following key management tasks that the mobile phone operating sys
out.

Process management ....

....

....

....

....

....

Memory management ....

....

....

....

....

....
[6]

(b) Leonardo uses the mobile phone to record his voice.

(i) Describe how sound sampling is used by the mobile phone to encode the sound.

....

....

....

.... [2]

(ii) Leonardo records his voice twice. Each recording is the same length a sampling resolution.

The first recording has a sampling rate of 44100 Hz. The second recordin sampling rate of 21000 Hz.

Describe how the different sampling rates will affect the recording and the sound file.

....

....

....

.... [2]

(iii) Leonardo transfers the recordings to his laptop computer. He uses sound editing software to delete some sections of the recordings, and copy and paste to replicate other sections.

Describe **two** other features of sound editing software Leonardo can use to edit the recordings.

1 ....

....

....

....

2 ....

....

....

....

[4]