

5 Seth uses a computer for work.

(a) Complete the following descriptions of internal components of a computer by writing the missing terms.

The ..... transmits the signals to coordinate events based on the electronic pulses of the .....

The ..... carries data to the components, while the

..... carries the address where data needs to be written to or read from.

The ..... performs mathematical operations and logical comparisons.

[5]

(b) Describe the ways in which the following factors can affect the performance of his laptop computer.

Number of cores

.....  
 .....  
 .....  
 .....

Clock speed

.....  
 .....  
 .....  
 ..... [4]

3 A processor has one general purpose register, the Accumulator (ACC), and several special purpose registers.

(a) Complete the following description of the role of the registers in the fetch-execute cycle by writing the missing registers.

The ..... holds the address of the next instruction to be loaded.

This address is sent to the .....

The ..... holds the data fetched from this address. This data is sent to the ..... and the Control Unit decodes the instruction's opcode.

The ..... is incremented. [5]

(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

<http://www.SirRazaAcademy.com>



(c) (i) The Accumulator currently contains the binary number:

(c) (i) The Accumulator currently contains the binary number:

0	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Write the contents of the Accumulator after the processor has executed the following instruction:

LSL #2

--	--	--	--	--	--	--	--

[1]

(ii) The Accumulator currently contains the binary number:

0	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Identify the mathematical operation that the following instruction will perform on the contents of the accumulator.

LSR #3

.....

..... [1]

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
	100

**ASCII code table (selected codes only)**

ASCII code	Character
65	A
66	B
67	C
68	D



(b) Some bit manipulation instructions are shown in the table:

Instruction		Explanation
Opcode	Operand	
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>
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>

<address> can be an absolute address or a symbolic address  
 # denotes a denary number, e.g. #123

The contents of the memory address 300 are shown:

Bit Number	7	6	5	4	3	2	1	0
300	0	1	1	0	0	1	1	0

(i) The contents of memory address 300 represent an unsigned binary integer.

Write the denary value of the unsigned binary integer in memory address 300.

..... [1]

(ii) An assembly language program needs to test if bit number 2 in memory address 300 is a 1.

Complete the assembly language instruction to perform this test.

..... #4 [1]

(iii) An assembly language program needs to set bit numbers 4, 5, 6 and 7 to 0, but keep bits 0 to 3 with their existing values.

Write the assembly language instruction to perform this action.

.....  
 ..... [2]



9608/11 Jun 18 Q8a, b

8 The Von Neumann model uses a series of registers. (a) Explain what is meant by the term register.

.....  
 .....  
 .....  
 .....[2]

(b) (i) Explain the purpose of the Memory Data Register (MDR).

.....  
 .....  
 .....  
 .....[2]

(ii) Name two registers, other than the MDR, that are used in the fetch-execute cycle.

Register 1 .....

Register 2 ..... [2]

9608/11 Jun 17 Q4a

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



PC .....

.....

.....

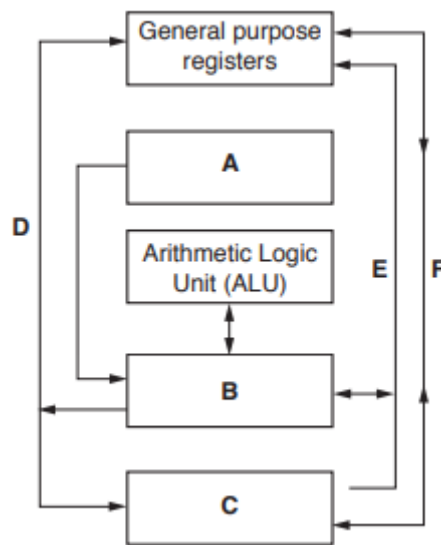
.....

.....

[4]

9608/11 Nov 17 Q4

4 (a) The diagram shows the components and buses found inside a typical Personal Computer (PC).



Some components and buses only have labels **A** to **F** to identify them. For each label, choose the appropriate title from the following list. The title for label **D** is already given.

- Control bus
- System clock
- Data bus
- Control unit
- Main memory
- Secondary storage

A .....

B .....

C .....

D Address bus

E .....

F ..... [5]

(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		Op code (binary)	Explanation
Op code (mnemonic)	Operand		
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:
CountDown:	OUTCH
	END
	15
	32
	51
Num1:	67
	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



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

9608/11 Nov 18 Q4a, b, d

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

Instruction		Explanation
Op code	Operand	
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 symbolic addressing or absolute addressing.

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

Instruction	Symbolic	Absolute
ADD 90		
CMP found		
STO 20		

[2]

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

X	1	0	1	1	1	0	1	0
---	---	---	---	---	---	---	---	---

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

Convert the value in X into denary.

..... [1]

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

Convert the value in X into hexadecimal.

..... [1]

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

Convert the value in X into denary.

..... [1]

Ch#1



(d) The current contents of the main memory, Index Register (IX) and selected values from the 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	/

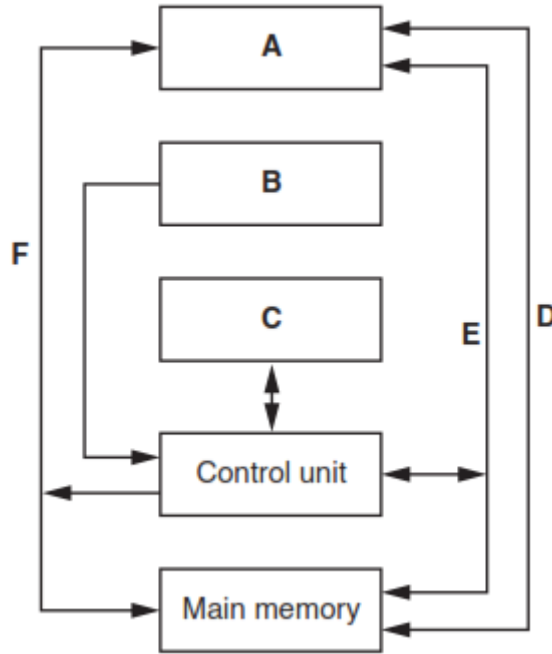
Instruction set

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.



9608/12 Nov 17 Q4

4 The following diagram shows the components and buses found inside a typical personal computer (PC).



(a) Some components and buses only have labels A to F to identify them.

For each label, choose the appropriate title from the following list. The title for label D is already given.

- Control bus
- Address bus
- Arithmetic Logic Unit (ALU)
- General purpose registers
- Secondary storage
- System clock

A .....

B .....

C .....

D Data bus

E .....

F ..... [5]

(b) Clock speed is a factor that affects the performance of a PC. Explain this statement.

.....  
 .....  
 ..... [2]

(c) An assembly language program can contain both **macros** and **directives**.

(i) Explain what is meant by these terms.

Macro .....

.....

.....

Directive .....

.....

.....[3]

(ii) Give an example of the use of a directive.

.....

.....[1]

(d) 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	
LDD	<address>	Direct addressing. Load the contents of the given address to ACC.
LDV	#n	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>	Store the contents of ACC at the given address.
INC		Increment the contents of ACC.
OUTCH		Output the character corresponding to the ASCII character code in ACC.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JMP	<address>	Jump to the given address.
CMP	#n	Compare the contents of ACC with number n.

Complete the trace table for the following assembly language program.

Label	Instruction
StartProg:	LDV #Offset
	CMP Value
	JPE EndProg
	OUTCH
	LDD Offset
	INC
	STO Offset
	JMP StartProg
EndProg:	END
Offset:	10
	50
	65
	89
	32
Value:	32

ASCII code table (selected codes only)				
<Space>	2	A	B	Y
32	50	65	66	89

Trace table:

ACC	Offset	OUTPUT
	10	
50		2
10		

[5]

(e) The program given in part (d) is to be translated using a two-pass assembler. The program has been copied here for you.

Label	Instruction
StartProg:	LDV #Offset
	CMP Value
	JPE EndProg
	OUTCH
	LDD Offset
	INC
	STO Offset
	JMP StartProg
EndProg:	END
Offset:	10
	50
	65
	89
	32
Value:	32

On the first pass, the assembly process adds entries to a symbol table.

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

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

Complete the symbol table. Use circular labels to show the order in which the assembler makes the entries.

Symbol table

Symbolic address	Relative address
StartProg (1)	0 (2)
Offset (3)	UNKNOWN (4)
Value (5)	

[6]

9608/12 Nov 18 Q3

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

Instruction		Explanation
Op code	Operand	
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 addressing**.

Indexed addressing .....

Immediate addressing .....[2]

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

Indexed addressing .....

Immediate addressing .....[2]

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



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

Convert the value in X into denary.

..... [1]

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

Convert the value in X into hexadecimal.


..... [1]

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

Convert the value in X into denary.

..... [1]

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

Address	Instruction
40	LDD 100
41	CMP 104
42	JPE 54
43	LDX 100
44	CMP 105
45	JPN 47
46	OUT
47	LDD 100
48	DEC ACC
49	STO 100
50	INC IX
51	JMP 41
52	
53	
54	END
...	
100	2
101	302
102	303
103	303
104	0
105	303
IX	1

ASCII code table (selected codes only)

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

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.

<http://www.SirRazaAcademy.com>



(a) Identify the line numbers of **three** errors that the student has made. Write the correct notation for each error.

Line number of error	Correct notation

[3]

(b) One stage of the FE cycle includes checking for interrupts.

(i) Give **three** different events that can generate an interrupt.

- 1 .....
- 2 .....
- 3 ..... [3]

(ii) Explain how interrupts are handled during the fetch-execute cycle.

.....  
 .....  
 .....  
 .....  
 .....  
 .....  
 .....  
 .....  
 .....  
 .....  
 ..... [5]

(c) The processor uses buses in the FE cycle. The diagram shows three buses and two descriptions. Draw **one** line from each bus to its appropriate description.



[2]

5 This question presents three scenarios. For each scenario, tick (✓) **one** box to show whether you think the person’s behaviour is ethical or unethical. Justify your choice.

(a) Wendy is a software engineer who is developing a program for her company. Her friend, Noah, is developing a program for a different company. Wendy looks at the code that Noah is writing to get ideas for her own program.

Ethical	
Unethical	

Justification .....

.....

.....

..... [2]

(b) Amit is fixing some bugs in the computer system of a large multinational company. He is asked to sign a confidentiality agreement. He sees some confidential information which contains the names of other multinational companies that have broken the law. He copies this information and releases it on the Internet.

Ethical	
Unethical	

Justification .....

.....

.....

..... [2]

(c) Farah is providing a company with an estimate for the cost of writing a program. The company she works for is in financial difficulty so she increases the estimate by 10%.

Ethical	
Unethical	

Justification .....

.....

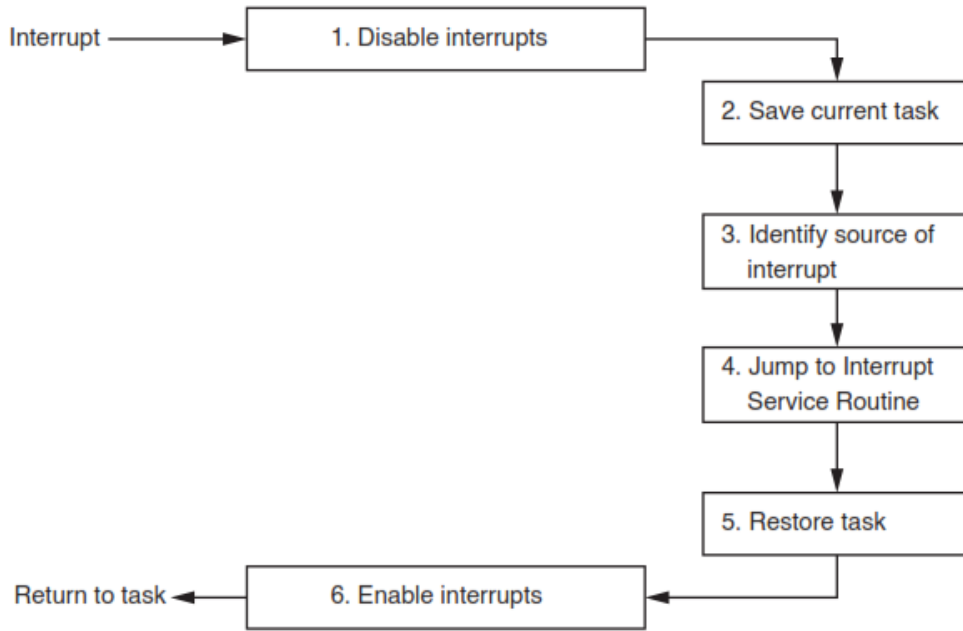
.....

..... [2]

9608/31 Jun 17 Q6c

(c) An alternative method of reading and processing sensor data is to use interrupts. Each sensor is connected so that it can send an interrupt signal to the processor if its value changes.

On receipt of an interrupt signal, the processor carries out a number of steps as shown in the following diagram.



(i) State the purpose of step 1.

.....  
 .....  
 .....[1]

(ii) State the purpose of step 6.

.....  
 .....  
 .....[1]

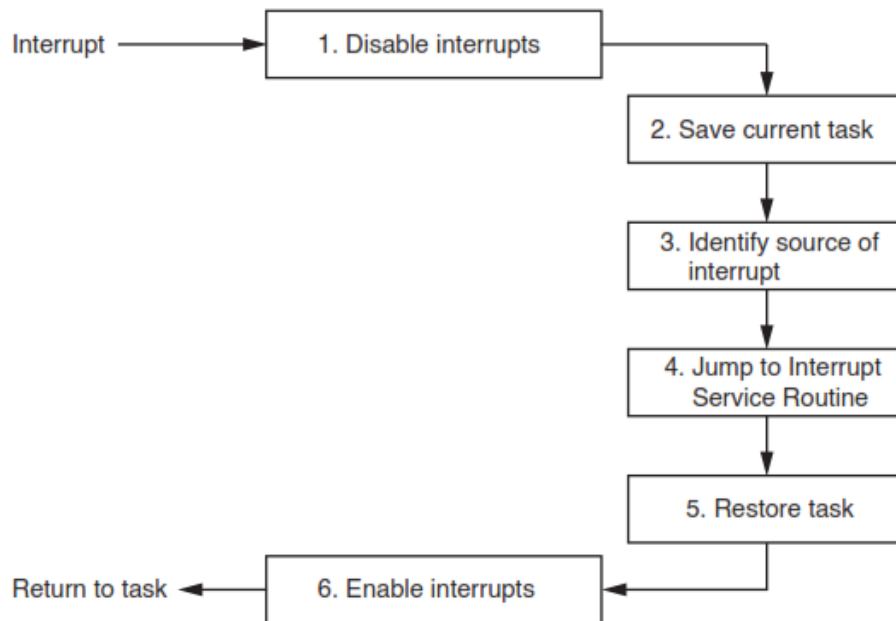
(iii) Explain how the current task is saved in step 2.

.....  
 .....  
 .....  
 .....[2]

9608/32 Jun 17 Q6c

(c) An alternative method of reading and processing sensor data is to use interrupts. Each sensor is connected so that it can send an interrupt signal to the processor if its value changes.

On receipt of an interrupt signal, the processor carries out a number of steps as shown in the following diagram.



(i) State the purpose of step 3.

.....  
 ..... [1]

(ii) Explain what happens at step 4.

.....  
 .....  
 ..... [2]

9608/12 Jun 17 Q5

5 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		Op code (binary)	Explanation
Op code (mnemonic)	Operand		
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]

9608/12 Nov 16 Q5

- 5 The table shows assembly language instructions for a processor that has one general purpose 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>	Index addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated 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.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to contents of the register (ACC or IX).
ADD	<address>	Add the contents of the given address to the ACC.
END		Return control to the operating system.

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

**Main memory**

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



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

LDD 102

ACC: 

--	--	--	--	--	--	--	--

[1]

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

LDX 101

IX: 

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

ACC: 

--	--	--	--	--	--	--	--

Explain how you arrived at your answer.

.....S.....  
 .....  
 .....  
 .....[2]

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

LDI 103

ACC: 

--	--	--	--	--	--	--	--

Explain how you arrived at your answer.

.....  
 .....  
 .....  
 .....  
 .....  
 .....[3]

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

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

Trace table:

ACC	Memory address			
	810	811	812	813
	28	41	0	0

[6]

h