SADIQ PUBLIC SCHOOL

Do the right, fear no man

Class: H2 Homework Worksheet

Saturday, 10th February 2024

Subject: Physics

Solve the Q2 of paper 5

2 A student is investigating a non-inverting operational amplifier (op-amp) circuit.

The circuit is set up as shown in Fig. 2.1.

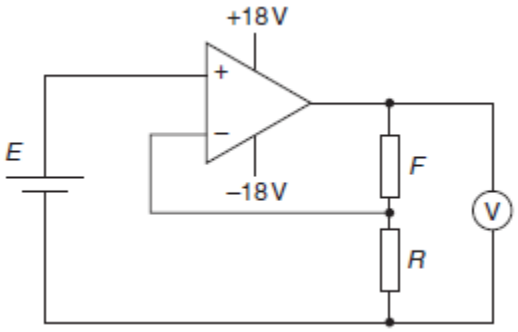


Fig. 2.1

The op-amp is connected to a +18 V and –18 V power supply.

E is the e.m.f. of the cell, which has a value of 1.6 ± 0.1 V.

An experiment is carried out to investigate how the reading *V* on the voltmeter varies with resistance *R*.

It is suggested that *V* and *R* are related by the equation

$$V = \frac{F}{R}E + E$$

where *F* is the resistance of the fixed resistor in the circuit.

(a) A graph is plotted of $\frac{V}{E}$ on the *y*-axis against $\frac{1}{R}$ on the *x*-axis. Express the gradient in terms of *F*.

gradient = ...[1]

(b) Values of *R* and *V* are given in Fig. 2.2.

R/Ω	V/V	$\frac{1}{R}/10^{-3}\Omega^{-1}$	$\frac{V}{E}$
150	14.4 ± 0.1		
220	10.4 ± 0.1		
330	7.4 ± 0.1		
470	5.6 ± 0.1		
680	4.4 ± 0.1		
860	3.8 ± 0.1		

Fig. 2.2

Calculate and record values of $\frac{1}{R}$ / $10^{-3}\,\Omega^{-1}$ and $\frac{V}{E}$ in Fig. 2.2. Include the absolute uncertainties in $\frac{V}{E}$. [3]

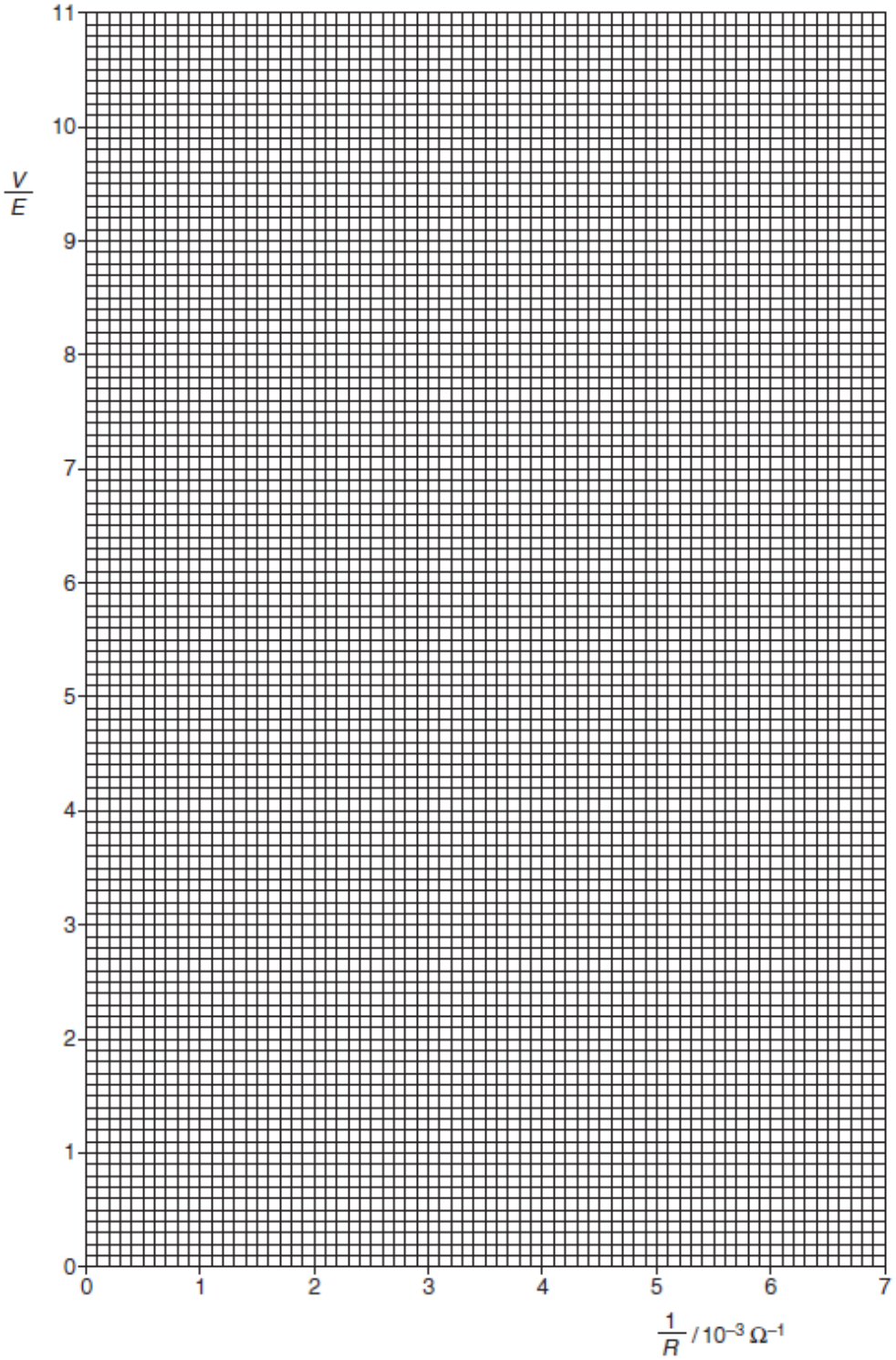
(c) (i) Plot a graph of $\frac{V}{E}$ against $\frac{1}{R}$ / $10^{-3}\,\Omega^{-1}$ Include error bars for $\frac{V}{E}$. [2]

(ii) Draw the straight line of best fit and a worst acceptable straight line on your graph.

Both lines should be clearly labelled. [2]

(iii) Determine the gradient of the line of best fit. Include the uncertainty in your answer.

gradient = ...[2]



(d) Using your answer in **(c)(iii)**, determine the value of *F*. Include the absolute uncertainty in your value and an appropriate unit.

F = ...[2]

(e) For one measurement, *R* has a value of 120 Ω ± 5%.

(i) Determine the value of $\frac{V}{E}$. using the relationship given and your answer in **(d)**. Include the absolute uncertainty in your answer.

$\frac{V}{E}$= .....[2]

(ii) Determine the expected voltmeter reading.

voltmeter reading = ... V [1]

Subject: Chemistry

Lesson5- Gas Liquid Chromatography

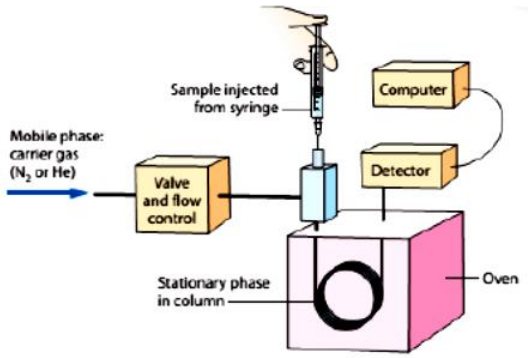
A: Inquiry

Gas chromatography (GC) with a gaseous mobile phase and a liquid. GC is primarily used for the identification of volatile compounds in environmental, medical, and forensic studies.

B: Information:

In GC, the mobile phase is a gas, not a liquid. The inert carrier gas, usually nitrogen or helium, flows through a column inside a heated oven. The inside of the column is coated with a thin layer of a stable, non-volatile liquid, such as silicone oil.

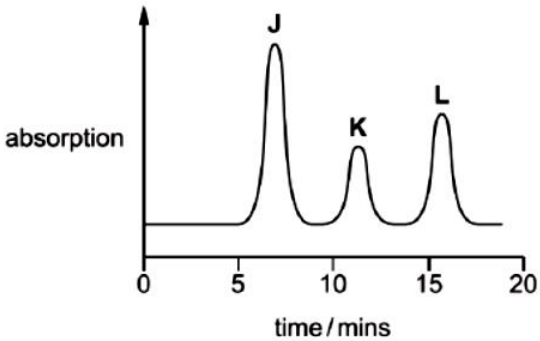
The sample being analysed could be a gas or a volatile liquid, which vaporises in the oven. When it is injected into the gas stream, the sample gets carried along through the column. The components are partitioned between the gas phase and the liquid in the column.



C. Practising

Gas Liquid Chromatography

1- (a) An alkene, a carboxylic acid and a ketone, all of similar volatility, are mixed together. The mixture is then analysed by gas chromatography.The gas chromatogram produced is shown.



The separation of the three compounds depends on their relative solubilities in the liquid stationary phase. The liquid stationary phase is an alkane.

(i) Complete the table to suggest which compound in the mixture is responsible for each peak J, K and L. Explain your answer by reference to the intermolecular forces of each compound.

peak	organic compound	explanation
J		
K		
L		

[2]

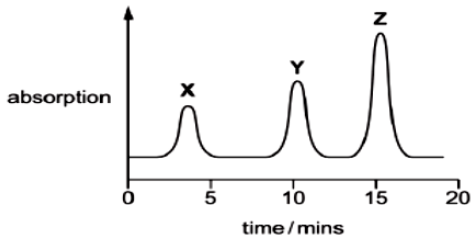
A student calculates the areas underneath the three peaks in the chromatogram.

peak	J	K	L
area / mm²	46	18	28

(ii) The area underneath each peak is proportional to the mass of the respective compound. Calculate the percentage by mass in the original mixture of the compound responsible for peak K.

% of mixture responsible for peak K= ..... [1]

2- (a)An aldehyde, an alkane and a carboxylic acid, all of similar volatility, are mixed together. The mixture is then analysed in a gas chromatograph. The gas chromatogram produced is shown.



The separation of the compounds depends on their relative solubilities in the stationary phase. The stationary phase is a liquid alcohol.

(i) Complete the table to suggest which compound in the mixture is responsible for each peak X, Yand Z. Explain your answer by reference to the intermolecular forces of the compounds.

peak	organic compound	explanation
X		
Y		
Z		

[2]

(ii) A student calculates the areas underneath the three peaks in the chromatogram.

peak	X	Y	Z
area / mm²	19	32	47

The area underneath each peak is proportional to the mass of the respective compound. Calculate the percentage by mass in the original mixture of the compound responsible for peak Z.

% of mixture responsible for peak Z= ..... [1]

Subject: Biology

Photosynthesis (Revision)

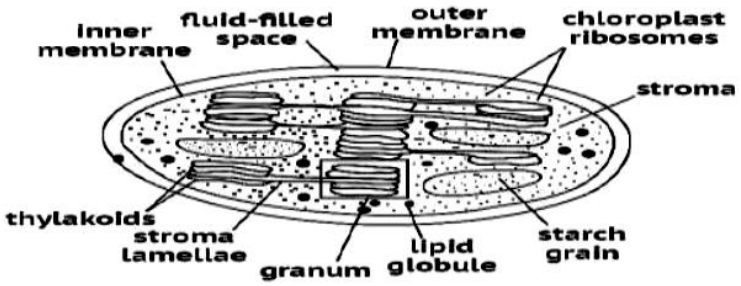
A) Inquiry:

Photosynthesis is a series of reactions in which energy transferred as light is transformed to chemical energy. Energy from light is trapped by chlorophyll, and this energy is then used to make carbohydrates Where do the cells of the plants get energy from for carrying out activities?

B) Information

Chloroplast

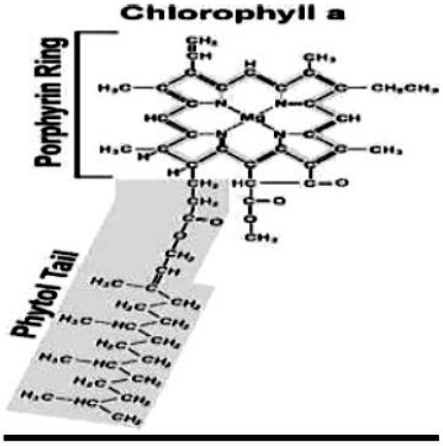
Photosynthesis takes place inside chloroplasts. These are organelles surrounded by two membranes, called an envelope. Below the membrane Thylakoids are present and their membrane are called lamellae, and it is here where the light-dependent reactions take place. The membranes contain chlorophyll molecules, arranged in groups called photosystems. There are two kinds of photosystems, PSI and PSII, each of which contains slightly different kinds of chlorophyll. Thylakoids are often arranged in stacks called grana (singular: granum).The liquid part of the chloroplast is called the stroma, and this is where the light-independent reactions take place. Chloroplasts often contain starch grains and lipid droplets. They are stores of energy-containing substances that have been made in the chloroplast but are not immediately needed by the cell or by other parts of the plant



Chlorophyll molecule structure

Its molecule contains 2 regions

- **Head porphyrin ring** which is similar to hemoglobin but in chlorophyll central atom is Mg^{+2} instead of Fe^{+2}.
- The second region is **Tail** which is long tail of hydrocarbons also known as **phytol** the hydrophobic tail is embedded in the bilayer of the thylakoid membrane where as head protrudes out.



Synthesising /absorbing the information

Write your own summary- notes in your notes book based on information you read in information section and what your book says about Photosynthesis and its steps.

C) Practising. (Read your text book for detailed information)

Q.1 Fig. 1.1 shows the changes in concentration of a 3C compound, glycerate phosphate, GP, and a 5C compound, ribulose bisphosphate, RuBP, extracted from samples taken from actively photosynthesising green algae in an experimental chamber when the light source was turned off.

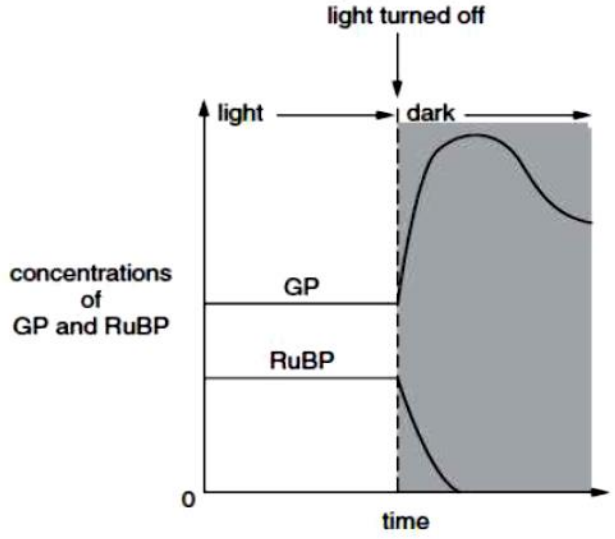


Fig. 1.1

(a) With reference to Fig. 1.1, describe what happens after the light source was turned off to the concentration of

(i) GP;

.....

.....

..... [2]

(ii) RuBP.

.....

..... [1]

(b) Explain, with reference to the Calvin cycle, the reasons for these observed changes in

(i) GP;

.....

.....

..... [2]

(ii) RuBP.

.....

.....

..... [2]

(c) State the two products of photophosphorylation that drive the Calvin cycle.

1. .....

2. ..... [2]

[Total : 9]

Q. 2 **(a)** An **absorption** spectrum is a graph of the absorption of different wavelengths of light by a photosynthetic pigment.
An **action** spectrum is a graph of the rate of photosynthesis at different wavelengths of light.
Fig. 7.1 shows the absorption spectra of chlorophyll a and chlorophyll b as well as an action spectrum.

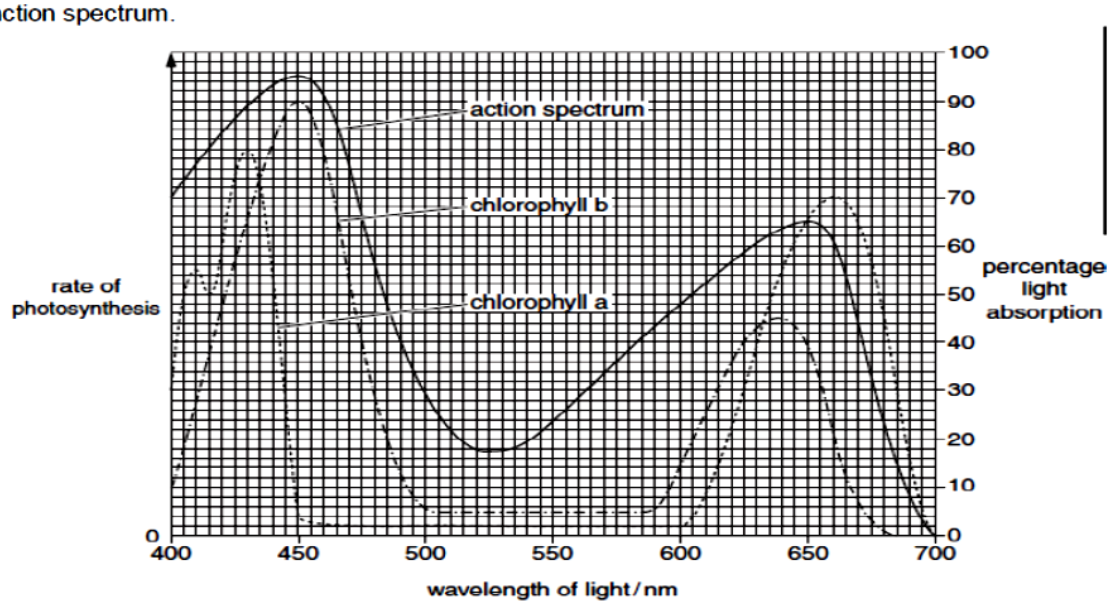


Fig. 7.1

With reference to Fig. 7.1,

(i) compare the **absorption** spectra of chlorophyll a and chlorophyll b,

.....

.....

.....

.....

.....

.....

..... [3]

(ii) explain the shape of the **action** spectrum,

.....

.....

.....

.....

..... [3]

(iii) explain why plants appear green .

.....

.....

.....

..... [2]

(b) Fig. 7.2 is an electron micrograph showing a section through part of a chloroplast.

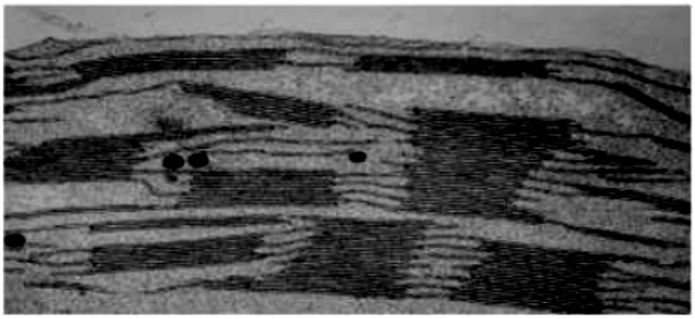


Fig. 7.2

On Fig. 7.2, draw label lines and use the letters **W** and **Y** to identify the following structures:

- **W** where the light-independent reactions occur
- **Y** where chlorophyll is found [2]

(b) Explain why increasing the concentration of carbon dioxide may increase the rate of production of carbohydrates at high light intensities.

.....

.....

.....

.....

.....

.....

..... [5]

[Total: 15]

Subject: Mathematics

1 When Don plays tennis, 65% of his first serves go into the correct area of the court. If the first serve goes into the correct area, his chance of winning the point is 90%. If his first serve does not go into the correct area, Don is allowed a second serve, and of these, 80% go into the correct area. If the second serve goes into the correct area, his chance of winning the point is 60%. If neither serve goes into the correct area, Don loses the point.

(i) Draw a tree diagram to represent this information. **[4]**

(ii) Using your tree diagram, find the probability that Don loses the point. **[3]**

(iii) Find the conditional probability that Don's first serve went into the correct area, given that he loses the point. **[2]**

2 When Andrea needs a taxi, she rings one of three taxi companies, A, B or C. 50% of her calls are to taxi company A, 30% to B and 20% to C. A taxi from company A arrives late 4% of the time, a taxi from company B arrives late 6% of the time and a taxi from company C arrives late 17% of the time.

(i) Find the probability that, when Andrea rings for a taxi, it arrives late. **[3]**

(ii) Given that Andrea's taxi arrives late, find the conditional probability that she rang company B. **[3]**

3 Data about employment for males and females in a small rural area are shown in the table.

	Unemployed	Employed
Male	206	412
Female	358	305

A person from this area is chosen at random. Let M be the event that the person is male and let E be the event that the person is employed.

(i) Find P(M). **[2]**

(ii) Find P(M and E). **[1]**

(iii) Are M and E independent events? Justify your answer. **[3]**

(iv) Given that the person chosen is unemployed, find the probability that the person is female. **[2]**

4 The probability that Henk goes swimming on any day is 0.2. On a day when he goes swimming, the probability that Henk has burgers for supper is 0.75. On a day when he does not go swimming the probability that he has burgers for supper is x. This information is shown on the following tree diagram.

Subject: Computer

Lesson: This lesson aims to revise the topic "Low Level Paradigms", through past paper solving practise.

Inquiry

What is do you know about assembly language? Which type of programming paradigm is used in Assembly Language code? Name some assembly language instructions you have used in your AS computer science course.

Information

Low-level programming refers to the development of software at a level close to the hardware and the architecture of the computer system. It involves working with instructions and representations that are closely tied to the hardware components. Low-level programming paradigms provide a high degree of control over the system's resources, but they often require a deep understanding of the hardware architecture. There are two primary types of low-level programming paradigms:

1). Machine Code and Assembly Language Programming:

Machine Code: Machine code consists of binary instructions that are directly executed by the computer's central processing unit (CPU). Each instruction corresponds to a specific operation at the hardware level.

Assembly Language: Assembly language is a human-readable, symbolic representation of machine code. It uses mnemonics to represent operations and addresses, making it easier for programmers to understand and write code. Assembly language is specific to a particular CPU architecture.

2). Embedded Systems Programming:

Embedded systems involve programming software for specialized computing devices, such as microcontrollers or custom hardware. In this paradigm, programmers often work with low-level languages, including Assembly, C, or a combination of both, to optimize code for the constrained resources of the embedded system.

Synthesising/Absorbing the information

- Revise your book contents pages 123 - 131
- Watch youtube video: https://youtu.be/wbZUKfM1fcA
- Add on important aspect discussed into your revision notes.

Practising

Attempt the following questions based on "Low Level Paradigm".

Q. 1.

(a) The numerical difference between the ASCII code of an upper case letter and the ASCII code of its lower case equivalent is 32 denary (32_{10}).
For example, 'F' has ASCII code 70 and 'f' has ASCII code 102.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0

The bit patterns differ only at bit number 5. This bit is 1 if the letter is lower case and 0 if the letter is upper case.

(i) A program needs a mask to ensure that a letter is in **upper case.**
Write the binary pattern of the mask in the space provided in the table below.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0
Mask								

Give the bit-wise operation that needs to be performed using the mask and the ASCII code. [2]

…….....

(ii) A program needs a mask to ensure that a letter is in **lower case**.
Write the binary pattern of the mask in the space provided in the table below.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0
Mask								

Give the bit-wise operation that needs to be performed using the mask and the ASCII code.

.....[2]

The following table shows part of the instruction set for a processor which has one general *purpose register, the Accumulator (ACC), and an index register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
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.
LDR	#n	Immediate addressing. Load the number n into IX.
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>.
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.
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>.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer is writing a program that will output the first character of a string in upper case and the remaining characters of the string in lower case.
The program will use locations from address WORD onwards to store the characters in the string.
The location with address LENGTH stores the number of characters that make up the string.

The programmer has started to write the program in the following table. The comment column contains descriptions for the missing program instructions.
(b) Complete the program using op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// initialise index register to zero
			// get first character of WORD
			// ensure it is in upper case using MASK1
			// output character to screen
			// increment index register
			// load 1 into ACC
			// store in COUNT
LOOP:			// load next character from indexed address WORD
			// make lower case using MASK2
			// output character to screen
			// increment COUNT starts here
			// is COUNT = LENGTH ?
			// if FALSE, jump to LOOP
			// end of program
COUNT:			
MASK1:			// bit pattern for upper case
MASK2:			// bit pattern for lower case
LENGTH:		4	
WORD:		B01100110	// ASCII code in binary for 'f'
		B01110010	// ASCII code in binary for 'r'
		B01000101	// ASCII code in binary for 'E'
		B01000100	// ASCII code in binary for 'D'

[12]

Q. 2.

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

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
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.
LDR	#n	Immediate addressing. Load the number n into IX.
STO	<address>	Store the contents of ACC at the given address.
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>.
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.

A programmer is writing a program that outputs a string, first in its original order and then in reverse order.

The program will use locations starting at address NAME to store the characters in the string. The location with address MAX stores the number of characters that make up the string.

The programmer has started to write the program in the table opposite. The Comment column contains descriptions for the missing program instructions.

Complete the program using op codes from the given instruction set. [15]

Label	Op code	Operand	Comment
START:			// initialise index register to zero
			// initialise COUNT to zero
LOOP1:			// load character from indexed address NAME
			// output character to screen
			// increment index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP1
REVERSE:			// decrement index register
			// set ACC to zero
			// store in COUNT
LOOP2:			// load character from indexed address NAME
			// output character to screen
			// decrement index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP2
			// end of program
COUNT:			
MAX:	4		
NAME:	B01000110		// ASCII code in binary for 'F'
	B01010010		// ASCII code in binary for 'R'
	B01000101		// ASCII code in binary for 'E'
	B01000100		// ASCII code in binary for 'D'

Q. 3.

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

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the 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.
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>.
JMP	<address>	Jump to the given 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.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <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.

(a) A programmer writes a program that:

- reads two characters input from the keyboard (you may assume they will be capital letters in ascending alphabetical sequence)
- outputs the alphabetical sequence of characters from the first to the second character.

For example, if the characters 'B' and 'F' are input, the output is:

BCDEF

The programmer has started to write the program in the following table. The Comment column contains descriptions for the missing program instructions, labels and data.
Complete the following program. Use op codes from the given instruction set. [9]

Label	Op code	Operand	Comment
START:			// INPUT character
			// store in CHAR1
			// INPUT character
			// store in CHAR2
			// initialise ACC to ASCII value of CHAR1
			// output contents of ACC
			// compare ACC with CHAR2
			// if equal jump to end of FOR loop
			// increment ACC
			// jump to LOOP
ENDFOR:	END		
CHAR1:			
CHAR2:			

(b) The programmer now starts to write a program that:
- converts a positive integer, stored at address NUMBER1, into its negative equivalent in two's complement form
- stores the result at address NUMBER2

Complete the following program. Use op codes from the given instruction set. [6]
Show the value stored in NUMBER2.

Label	Op code	Operand	Comment
START:			
		MASK	// convert to one's complement
			// convert to two's complement
	END		
MASK:			// show value of mask in binary here
NUMBER1:	B00000101		// positive integer
NUMBER2:			// negative equivalent

Q. 4.

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

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the 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.
STO	<address>	Store the contents of ACC at the given address.
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>.
JMP	<address>	Jump to given 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.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <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.

(a) A programmer writes a program that:

- reads a character from the keyboard (assume it will be a capital letter)
- outputs the alphabetical sequence of characters from 'A' to the character input. For example, if the character 'G' is input, the output is:

ABCDEFG

The programmer has started to write the program in the table on the following page. The Comment column contains descriptions for the missing instructions, labels and data.

Complete the following program. Use op codes from the given instruction set. [8]

Label	Op code	Operand	Comment
START:			// INPUT character
			// store in CHAR
			// Initialise ACC (ASCII value for 'A' is 65)
			// OUTPUT ACC
			// compare ACC with CHAR
			// if equal jump to end of FOR loop
			// increment ACC
			// jump to LOOP
ENDFOR:	END		
CHAR:			

(b) The programmer now starts to write a program that:

- tests whether an 8-bit two's complement integer stored at address NUMBER is positive or negative
- outputs 'P' for a positive integer and 'N' for a negative integer.

Complete the following program. Use op codes from the given instruction set.
Show the required value of MASK in binary. [7]

Label	Op code	Operand	Comment
START:			
		MASK	// set to zero all bits except sign bit
			// compare with 0
			// if not equal jump to ELSE
THEN:			// load ACC with 'P' (ASCII value 80)
	JMP	ENDIF	
ELSE:			// load ACC with 'N' (ASCII value 78)
ENDIF:			
	END		
NUMBER:	B00000101		// integer to be tested
MASK:			// value of mask in binary

Q. 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		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.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated 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.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
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.

(a) Six letters are stored, starting at the location labelled LETTERS. A program is needed to perform a linear search on LETTERS to find the letter 'x'. The program counts the number of times 'x' appears in LETTERS.
The following is the pseudocode for the program.

```
FOR COUNT 0 TO 5
  IF LETTERS[COUNT] = LETTERTOFIND
  THEN
      FOUND FOUND + 1
  ENDIF
ENDFOR
```

Write this program. Use the op codes from the given instruction set. [10]

Label	Op code	Operand	Comment
START:	LDR	#0	// initialise Index Register
LOOP:			// load LETTERS
			// is LETTERS = LETTERTOFIND ?
			// if not, go to NOTFOUND
			// increment FOUND
NOTFOUND:			// increment COUNT
			// is COUNT = 6 ?
			// if yes, end
			// increment Index Register
			// go back to beginning of loop
ENDP:	END		// end program
LETTERTOFIND:		'x'	
LETTERS:		'd'	
		'u'	
		'p'	
		'l'	
		'e'	
		'x'	
COUNT:		0	
FOUND:		0	

(b) Six values are stored, starting at the location VALUES. A program is needed to divide each of the values by 8 and store them back in their original location.
Write this program. Use the op codes from the instruction set on the next page. [10]

Label	Op code	Operand	Comment
START:			// initialise the Index Register
			// load the value from VALUES
			// divide by 8
			// store the new value in VALUES
			// increment the Index Register
			// increment REPS
			// is REPS = 6 ?
			// repeat for next value
	END		
REPS:	0		
VALUES:	22		
	13		
	5		
	46		
	12		
	33		

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.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated 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.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
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.

Q. 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
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.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated 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.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
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.

(a) A program stores a letter. The user is allowed nine attempts to guess the stored letter. The program outputs "?" and the user guesses a letter. If the user guesses the letter, the program outputs "*".
The following is pseudocode for this program.

```
REPEAT
    OUTPUT '?'
    INPUT GUESS
    IF GUESS = LETTERTOGUESS  THEN
        OUTPUT '*'
        BREAK
    ELSE
        ATTEMPTS ATTEMPTS + 1
    ENDIF
UNTIL ATTEMPTS = 9
```

Write this program. Use the op codes from the instruction set provided. [11]

Label	Op code	Operand	Comment
START:	LDM	#63	// load ASCII value for '?'
			// OUTPUT '?'
			// input GUESS
			// compare with stored letter
			// if correct guess, go to GUESSED
			// increment ATTEMPTS
			// is ATTEMPTS = 9 ?
			// if out of guesses, go to ENDP
			// go back to beginning of loop
GUESSED:	LDM	#42	// load ASCII for '*'
			// OUTPUT '*'
ENDP:	END		// end program
ATTEMPTS:	0		
LETTERTOGUESS:	'a'		

(b) Five numbers are stored, starting in the location labelled NUMBERS. A program is needed to multiply each of the numbers by 4 and store them back in their original location.
Write this program. Use the op codes from the instruction set on the next page. [10]

Label	Op code	Operand	Comment
START:			// initialise the Index Register
			// load the value from NUMBERS
			// multiply by 4
			// store the new value in NUMBERS
			// increment the Index Register
			// increment COUNT
			// is COUNT = 5 ?
			// repeat for next number
ENDP:	END		
COUNT:	0		
NUMBERS:	22		
	13		
	5		
	46		
	12		

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.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated 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.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
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.

Q. 7.

A train cannot move if any of the eight automatic train doors are open. The train door monitoring system, set out below, checks that all the doors are closed before the train can move.

- If a monitoring system detects that a door is open, it sets a specific bit in address 500 to 1.

- If the bit for door one is equal to 1, the binary value for hexadecimal FF is sent to address 501. The contents of address 501 are changed to make door 1's light flash when the door is open.

- If the bit for door two is equal to 1, the binary value for hexadecimal FF is sent to address 502. The contents of address 502 are changed to make door 2's light flash when the door is open. This is repeated for each door from 3 to 8.

- Each door sets its bit in address 500 to zero when the door closes, and the contents of the corresponding door address are set to zero.

- The train manager can identify which door is open from the flashing light.

The current contents of address 500 are:

	Door number							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Address 500	1	0	0	1	0	0	1	0

(a) Complete the following table by writing the values stored in addresses 503 to 508. Use the contents of address 500 shown above. Note that addresses 501 and 502 are complete. [2]

501	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	Door 1
502	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	Door 2
503									Door 3
504									Door 4
505									Door 5
506									Door 6
507									Door 7
508									Door 8

(b) The following table shows assembly language instructions for the processor controlling the train door monitoring system that has one general purpose register, the Accumulator (ACC).

	Instruction		Explanation
Label	**Op code**	**Operand**	
	LDM	&n	Load the hexadecimal number n to ACC
	LDD	<address>	Load the contents of the location at the given address to ACC
	STO	<address>	Store the contents of ACC at the given address
	AND	&n	Bitwise AND the contents of ACC with the hexadecimal number n
	CMP	&n	Compare the contents of ACC with the hexadecimal number n
	JPE	<address>	Following a compare instruction, jump to <address> or <label> if the compare was True
<label>:	<op code>	<operand>	Labels an instruction
	WAIT		Macro to wait one second before the next instruction is executed

After rechecking the doors, address 500 now contains 10101010.

(i) Complete the table by writing the values of the Accumulator (ACC) and the contents of address 501 as these instructions are executed **once** to check door **1**.

[4]

	Instruction		ACC	501
Label	**Op code**	**Operand**		
CHECK1:	LDD	500		
	AND	&80		
	CMP	&00		
	JPE	DOOR1		
	LDM	&FF		
DOOR1:	STO	501		
	WAIT			
	LDM	&00		
	STO	501		
	WAIT			
	JMP	CHECK1		

(ii) Write the assembly language instructions to check door **2**. [4]

	Instruction	
Label	Op code	Operand

(c) Explain how the check door routines show a flashing light or no light. [2]

.....

.....

.....

.....

Q. 8.

A company has a number of lorries that deliver items around the country. The items in each lorry are its load. Each lorry has a monitoring system that provides information to the driver about the state of the load and other data from each trip.

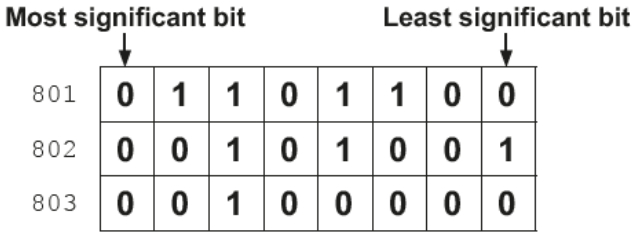
- Data is stored in three memory locations with addresses 801 to 803.
- Location 801 contains the distance travelled in kilometres for the current trip, stored as a binary integer.
- Location 802 contains the quantity of fuel used in litres for the current trip, stored as a fixed-point binary number with six places before the binary point and two places after the binary point.
- The four most significant bits of location 803 are flags used to identify problems with the load, for example it is too heavy. A flag is set to 1 if there is a problem, or 0 if not.

The problems are:

- Bit 7 – load too heavy
- Bit 6 – load too high

• Bit 5 – load unstable
• Bit 4 – load not secured (risk of the load falling off)
• Bits 0 to 3 are not used

(a) The current contents of addresses 801 to 803 are:



State the information that the current contents of addresses 801 to 803 will provide to the driver. [3]

.....

.....

.....

.....

(b) A lorry has a load that is too heavy and is not secured. It has travelled 120 kilometres and used 35.25 litres of fuel.
Complete the contents of the addresses to record this information. [3]

801								
802								
803								

(c) The following table shows the instructions for the lorry load monitoring system in assembly language. There is one general purpose register, the Accumulator (ACC).

Table 7.1

Instruction			Explanation
Label	**Op code**	**Operand**	
	LDM	#n	Load the number n to ACC
	LDD	<address>	Load the contents of the location at the given address to ACC
	STO	<address>	Store the contents of ACC at the given address
	AND	#n	Bitwise AND operation of the contents of ACC with the operand
	CMP	#n	Compare the contents of ACC with number n
	JPE	<address>	Following a compare instruction, jump to <address> or <label> if the compare was True
	JMP	<address>	Jump to the given address or label
<label>:	<op code>	<operand>	Labels an instruction

Note:

# denotes immediate addressing
B denotes a binary number, for example B01001010
& denotes a hexadecimal number, for example &4A

(i) Write **assembly language** instructions to set the contents of addresses 801 and 802 to zero, and set all four most significant bits of the contents of address 803 to one.
Use the instruction set from **Table** **[3]**

.....

.....

.....

.....

(ii) A program written in assembly language, continuously checks the flags. If a flag is set, the program jumps to the error-handling routine at the specified label. For example, if the load is too heavy, the program jumps to the error-handling routine with the label TOOHEAVY. The error-handling routine instructions have not been provided.
A programmer has written most of the instructions for the program in the following table. There are four missing operands.
Complete the assembly language program by writing the **four** missing operands. [4]

Label	Op code	Operand
CHECKLOAD:	LDD	803
	AND	&F0
	STO	TEMP
	AND	&80
	CMP	&80
	JPE	TOOHEAVY
	LDD	TEMP
	AND	&40
	CMP	
	JPE	TOOHIGH
	LDD	TEMP
	AND	
	CMP	&20
	JPE	UNSTABLE
	LDD	
	AND	&10
	CMP	&10
	JPE	NOTSECURED
	JMP	
TEMP:		

Q. 9.

A car monitoring system provides information to the driver about the car's performance and alerts the driver to possible problems.

- Data about the car's performance is stored in three memory locations with addresses 601 to 603.
- Location 601 contains the distance travelled in kilometres for the current trip as a binary integer.
- Location 602 contains the quantity of fuel used in litres for the current trip, as a fixed-point binary number with 5 places before the binary point and three places after the binary point.
- The four least significant bits of location 603 are flags used to identify problems with the car, for example, the fuel is low. A flag is set to 1 if there is a problem, or 0 if not. These problems are:
- Bit 0 - high engine temperature
- Bit 1 - low oil pressure
- Bit 2 - low battery
- Bit 3 - low fuel
- Bits 4 to 7 are not used

(a) The current contents of addresses 601 to 603 are:

	Most significant							Least significant
601	0	0	1	0	1	1	0	0
602	0	0	1	0	1	0	0	1
603	0	0	0	0	0	1	0	0

State the information that the current contents of addresses 601 to 603 will provide to the driver.

.....

.....

..... ……[3]

(b) A car has low oil pressure and low fuel. It has travelled 80 kilometres and used 7.25 litres of fuel. Complete the contents of the addresses to record this information. [3]

601								
602								
603								

(c) The following table shows the assembly language instructions for the car performance monitoring system. There is one general purpose register, the Accumulator (ACC).

Table 8.1

Instruction			Explanation
Label	**Op code**	**Operand**	
	LDM	#n	Load the number n to ACC
	LDD	<address>	Load the contents of the location at the given address to ACC
	STO	<address>	Store the contents of ACC at the given address
	AND	#n	Bitwise AND operation of the contents of ACC with the numeric operand
	CMP	#n	Compare the contents of ACC with the number n
	JPE	<address>	Following a compare instruction, jump to <address> or <label> if the compare was True
	JMP	<address>	Jump to <address> or <label>
<label>:	<op code>	<operand>	Labels an instruction

Note:

# denotes immediate addressing
B denotes a binary number, for example B01001010
& denotes a hexadecimal number, for example &4A

(i) Write **assembly language** instructions to set the contents of addresses 601 and 602 to zero, and set all four least significant bits of the contents of address 603 to one.
Use the instruction set from **Table** **[3]**

.....

.....

.....

.....

(ii) A program continuously checks the flags. If a flag is set, the program moves to the error-handling routine at the specified label. For example, if the engine temperature is high, the program jumps to the label for the error-handling routine HIGHTEMP. The error-handling routine instructions have not been provided.
A programmer has written most of the instructions for the program in the following table. There are four missing operands.

Complete the assembly language program by writing the **four** missing operands.

Label	Op code	Operand
CHECKFLAGS:	LDD	603
	AND	&0F
	STO	TEMP
	AND	&01
	CMP	&01
	JPE	HIGHTEMP
	LDD	TEMP
	AND	&02
	CMP	
	JPE	LOWOIL
	LDD	TEMP
	AND	
	CMP	&04
	JPE	LOWBATT
	LDD	
	AND	&08
	CMP	&08
	JPE	LOWFUEL
	JMP	
TEMP:		

Q. 10.

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

	Instruction		Explanation
Label	**Opcode**	**Operand**	
	LDM	#n	Load the number n to ACC
	LDD	<address>	Load the contents of the given address to ACC
	LDI	<address>	The address to be used is at the given address Load the contents of this second address to ACC
	ADD	<address>	Add the contents of the given address to the ACC
	STO	<address>	Store the contents of the ACC at the given address
<label>:		<data>	Gives a symbolic address <label> to the memory location with the contents <data> <label> can be used in place of <address>
# denotes a denary number, e.g. #123			

(a) The address 500 contains the value 100 and the address 100 contains the value 20.
State the addressing mode and the contents of ACC after each instruction has been executed. [3]

LDM #500 Addressing mode .....

Contents of ACC .....

LDD 500 Addressing mode .....

Contents of ACC .....

LDI 500 Addressing mode .....

Contents of ACC .....

(b) Use only the given instruction set to write **assembly language** code to: [7]

- use the constant 20 which needs to be stored
- add this constant to the value stored in the address contained in the variable Y
- store the result in variable Z.

Label	Instruction	
	Opcode	Operand

Subject: Business Studies

Topic: Analysis of published accounts

Reinforce the concept of ratios analysis through text book and lecture given in class and attempt following questions.

Selecting a company to invest in:

Q1. Use the internet and newspaper stock exchange prices pages to select two companies that you might be interested in investing in.

Q2. Calculate the current P/E ratios, dividend yield and dividend cover ratios. Also try to find out what they were one year ago.

Q3. Imagine you have 1000 units of your own currency to spend on buying the shares of one of these companies. Calculate how many shares you could buy.

Q4. Monitor carefully what happens to the share price of your chosen company over the next three months. Recalculate the three ratios above when more up-to-date data is available.

Q5. Was your investment in the shares of your company a good one? Explain your answer.

Subject: Accounting

Computerized Accounting System

Inquiry

Do you know that computerized accounting system has speeded up recording and reporting process? What is a transition period? What are the advantages and limitations of software? What is customized software? And what gadgets are required to work smoothly in account department?

Information

(i) **A computerized accounting system** is an accounting information system that processes the financial transactions and events as per Generally Accepted Accounting Principles (GAAP) to produce reports as per user requirements. ... First, it has to work under a set of well-defined concepts called accounting principles.

(ii) **Benefits** of computerized Accounting Systems

Reduce the time spent on manual processes.
Less errors and increased accuracy.
Real-time financial information.
Automated invoices, credit notes and receipts.
Innovative financial technology.
Save money on resources.

(iii) Read form your text book. Page no. 445-447

(iv) Watch this short video that shows a summary of computerised accounting: https://youtu.be/kZvHRmgS3W8

Synthesising /Absorbing

Students write your own summary- notes in your notes book based on information given above, read pages and watched video.

Practice

Solve three activities 1, 2 and 3 which are given on page no. 446.

Subject: Economics

Wage determination in imperfect markets

Inquiry:

What is meant by imperfect market?

How wages are determined under imperfect competition?

How do trade unions affect the labour market?

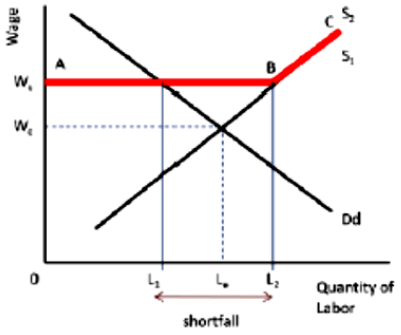
What is monopsony?

Information: Wage determination in imperfect markets

A labour market may not necessarily operate as a perfect market. For example, the workers may be members of a trade union and trade unions may therefore have an influence on the process of wage determination. It is also possible that a government may decide to intervene in the process. It may also be the case that there exists a single employer of labour, known as a monopsony.

Influence of trade unions on wage determination

It may be the case that workers belong to a trade union which is involved in collective bargaining with employers on behalf of the workers. A trade union may insist on the existence of a closed shop to increase its bargaining power with employers.



The trade union can reduce the supply of labourfor example, by pressurising the government and/or employers into making entry into particular employment more difficult and this will have an effect on wages.

Figure shows the demand (Dd) and the supply (S1) curve in a perfectly competitive market where wage is We and workers employed are Le. Power of trade unions enables them to negotiate wages above the equilibrium wage rate (Wu) in a competitive market.

This leads to a shift in the supply curve. Employers in the industry are not able to hire workers below Wu and the supply curve becomes perfectly elastic till point B. Beyond point B, employers will have to pay higher wage rates if they wanted to employ more workers leading to an upward sloping supply curve. The new supply curve after union

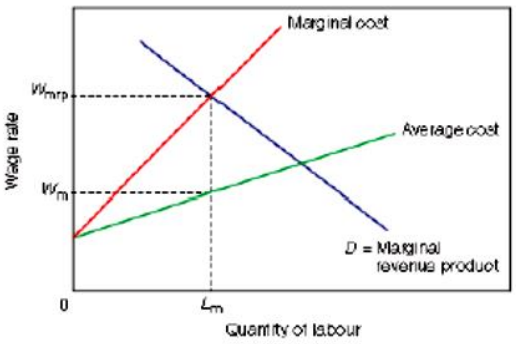
negotiated wages will be ABC.
The new equilibrium wage rate is Wu at which the demand for workers falls to L1 and their supply increases to L2. This leads to a shortfall equalling L2 – L1.
There is a danger that these unemployed people will undercut the union wage, unless union can limit supply by preventing firms from employing non-unionized labour. A good example of a successful trade union is the US Actors Guild. Guild is able to restrict the number of actors and actresses who are able to work in films, TV and theatre. This increases the bargaining power of the union enabling them to successfully negotiate higher wages.
Unions can only succeed in increasing wage without a fall in the level of employment if they agree to a productivity deal, i.e. the productivity of labour is increased.

Trade union intervention is therefore useful for those workers who remain employed as they get higher wages buy it is harmful for workers who lose their jobs. If the union is more concerned with securing higher wage rate, then it will have to accept some reduction in employment. And if it wants to maximize employment, then it will have to accept the market wage rate unless productivity deals can be negotiated.

Monopsony

A monopsony is a situation when a single buyer dominates a market. A monopsonist can act as a price maker, and drive down prices. For example, supermarkets are sometimes accused of acting as monopsonists when buying from their suppliers. Some people claim supermarkets unfairly use their market power to force suppliers to sell their products at a price that means those suppliers make a loss.

Figure shows how the monopsonist can affect the market equilibrium. The monopsonist will hire workers by equating the marginal cost paid to employ a worker with the marginal revenue product gained from this employment. This is the profit-maximising position. The wage that the monopsonist pays to hire labour is Wm. This is actually below the wage that should be paid if they were paying the full value of their marginal revenue product that is Wmrp. The level of employment is Lm.



In this situation the power of the employer in the labour market is of overriding importance and the employer can set a low wage because of this buying power. Monopsonists often exist in local labour market situations, for example where there is just one major employer in a town or where workers may be employed in an extractive industry located well away from where they normally live. In this way the employer dominates the labour market, setting down wages and all other conditions of employment.

Please read what your textbook says about Wage determination in imperfect markets

Please watch this brief YouTube video on monopsony
(https://www.youtube.com/watch?v=LNR7IbJktNE)

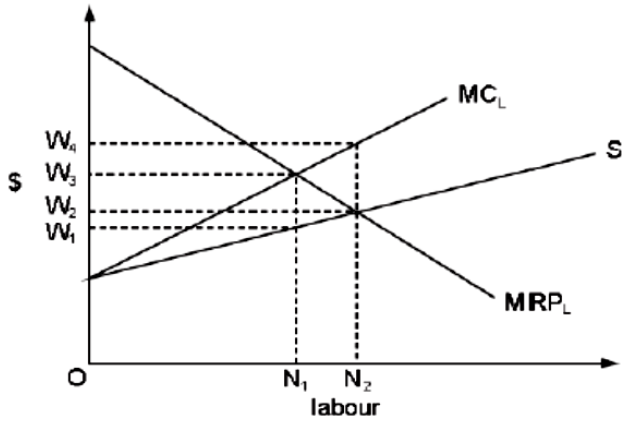
Synthesising/absorbing the information:

Write your own summary notes in your note book based on the information you read in your textbook and what you saw on the youtube video.

Practising:

MCQs

1 In the diagram, MRPL is a firm"s marginal revenue product of labour curve, S is its supply of labour curve, and MCL its marginal cost of labour curve.



Assuming profit maximisation, how many workers will the firm employ and what wage will it pay?

	number employed	wage
A	N_1	W_3
B	N_1	W_1
C	N_2	W_2
D	N_2	W_4

2 The workers in a firm have not previously belonged to a trade union but now join one.
In which circumstance is this most likely to raise their wages?
A Capital is highly substitutable for labour.
B Labour costs constitute a large fraction of the firm's costs.
C The demand for the firm's product is price-elastic.
D The firm has enjoyed monopsony power in the market for its labour.

Q In some countries the power of trade unions has decreased. In other countries, trade unions have organised major strikes resulting in employees refusing to work.

Is the existence of a trade union likely to be the main factor that affects the supply of labour?

Subject: Law

Answer essay question no 1 and 2 of May/June 2024 paper 32