**Week 42**

# A-Level Computer Science

Homework bundle for this week

本周作业合集

exercises.lol

## Contents 目录

**Topic 15 quiz** . . . . . 2

**Exercise sheet 15.1** . . . . . 5

**Past-paper questions 15.1** . . . . . 10

**Exercise sheet 15.2** . . . . . 11

**Past-paper questions 15.2** . . . . . 17

# 15. Hardware and Virtual Machines

**Starter Quiz · 课前小测**

A-Level Computer Science  Name: _____  Score: _____

1. A RISC processor is characterised by:

   ☐ a small set of simple, fixed-length instructions

   ☐ many complex variable-length instructions

   ☐ no registers

   ☐ instructions that each take many cycles

2. SIMD means:

   ☐ one instruction operates on many data items at once

   ☐ many instructions on one data item

   ☐ one instruction on one data item

   ☐ no instructions at all

3. By De Morgan's law, $\overline{A \cdot B}$ equals:

   ☐ $\overline{A} + \overline{B}$

   ☐ $\overline{A} \cdot \overline{B}$

   ☐ $A + B$

   ☐ $A \cdot B$

4. In a half adder, the sum output S is produced by which gate?

   ☐ XOR

   ☐ AND

   ☐ OR

   ☐ NOT

5. When a pipeline is full, it completes about:

   ☐ one instruction per clock cycle

   ☐ one instruction every five cycles

   ☐ all instructions at once

   ☐ no instructions

6. A data hazard stalls the pipeline when an instruction needs a result that is not ready yet; a control hazard comes from a branch changing which instruction runs next.

   ☐ True  ☐ False

7. A system VM (managed by a hypervisor) runs a whole guest operating system, whereas a process VM like the JVM just runs one program's portable bytecode.

   ☐ True ☐ False

8. In a Karnaugh map, a larger group of adjacent 1s eliminates more variables, giving a simpler term (a group of 2 drops one variable, a group of 4 drops two).

   ☐ True ☐ False

9. A flip-flop is bistable — it has two stable states and remembers one bit — which makes it the building block of registers and SRAM.

   ☐ True ☐ False

10. A massively parallel computer uses:

    ☐ thousands of processors with their own memory, exchanging messages

    ☐ a single very fast core

    ☐ one processor and a huge disk

    ☐ no network

# 15. Hardware and Virtual Machines

**Answers · 答案**

A-Level Computer Science

1. **a small set of simple, fixed-length instructions**

   RISC keeps instructions few, simple and fixed-length (usually 1 cycle); CISC has many complex variable-length ones.

2. **one instruction operates on many data items at once**

   Single Instruction, Multiple Data — e.g. a GPU running the same operation on thousands of pixels.

3. $\overline{A} + \overline{B}$

   Negate the whole, swap AND→OR, negate each operand: $\overline{A \cdot B} = \overline{A} + \overline{B}$.

4. **XOR**

   $S = A$ XOR $B$ (1 when the inputs differ); the carry is $A$ AND $B$.

5. **one instruction per clock cycle**

   Overlapping the stages means a new instruction finishes each cycle once the pipeline is full.

6. **True**

   Hazards force the pipeline to stall (or flush), which is why they reduce the ideal one-per-cycle throughput.

7. **True**

   System VMs share one physical machine among several guest OSes; process VMs give "write once, run anywhere" for a single program.

8. **True**

   You group adjacent 1s into power-of-two rectangles in Gray-code order; the bigger the group, the simpler the term it becomes.

9. **True**

   Chaining flip-flops gives registers and counters; SRAM cache is built from them (no refresh needed, unlike DRAM).

10. **thousands of processors with their own memory, exchanging messages**

    Massively parallel systems (supercomputers) link thousands of processors with distributed memory over a fast network.

A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

# 15.1 Processors, Parallel Processing and Virtual Machines

Name: _____ Class: _____ Date: _____ **Total: 17 marks**

**KEY WORDS** risc 精简指令集 • cisc 复杂指令集 • virtual machine 虚拟机 • mimd 多指令多数据 • simd 单指令多数据

## Objective

Build the skills to answer exam questions on **processors and parallelism** — RISC/CISC, pipelining, Flynn's taxonomy and virtual machines.

**You must be able to:**

- compare **RISC** and **CISC** processors
- explain **pipelining** and the role of **registers**
- describe **Flynn's taxonomy** and a **virtual machine**

## 1 Worked examples

Study these first. Each one shows the method for a question type used later — follow the steps and you can do the Practice and Exam-style questions yourself.

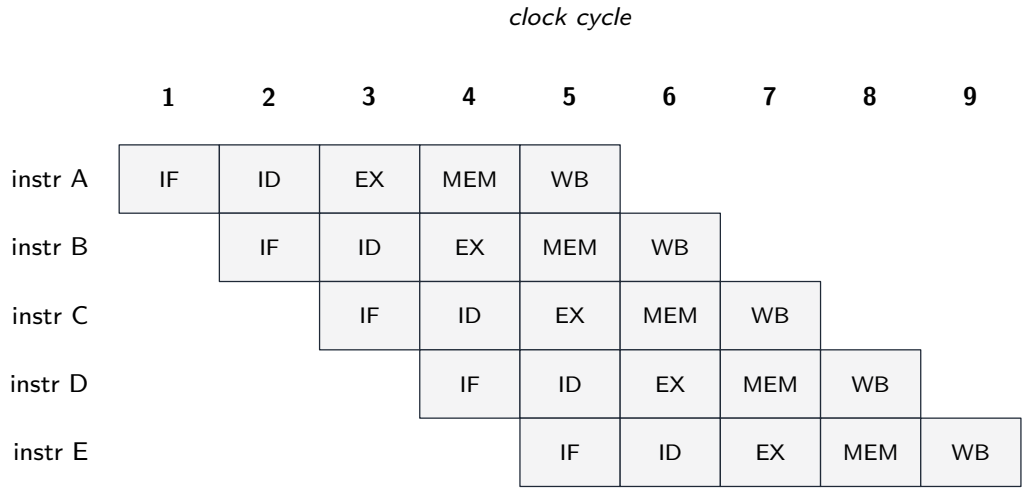
### ■ RISC vs CISC

| Feature | CISC | RISC |
|---|---|---|
| Instruction set | many, complex | few, simple |
| Instruction length | variable | fixed |
| Memory access | many instructions | only **load/store** |
| Pipeline-friendly | harder | naturally |

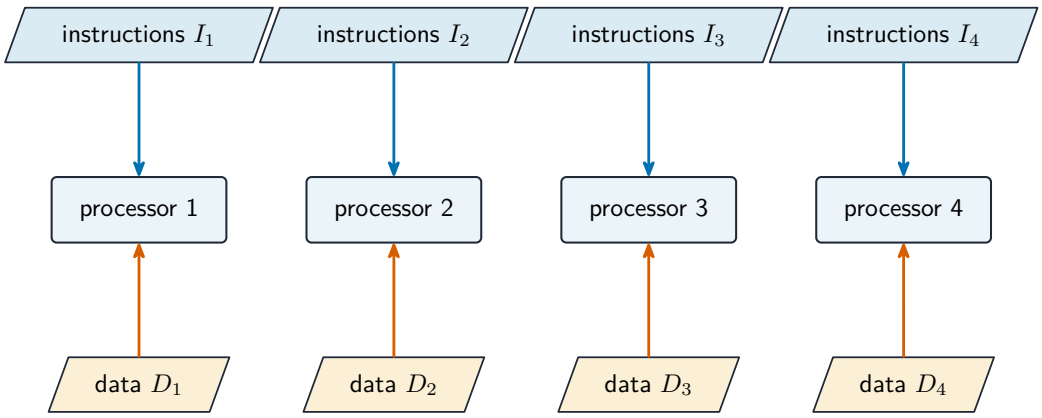
RISC keeps data in **many registers** (fast) and only `load`/`store` touch memory.

### ■ Pipelining

A **pipeline** overlaps the stages of consecutive instructions, like an assembly line, so once full, **one instruction completes per cycle**:




*pipeline full — one instruction finishes per cycle*




*MIMD parallelism runs many instruction streams*

RISC's **fixed-length, simple** instructions make each stage take equal time. A pipeline can **stall** on a **hazard** — a data hazard (a needed result isn't ready) or a control hazard (a branch).

## Flynn's taxonomy and virtual machines

- **SISD** — one instruction, one data stream (single core).
- **SIMD** — one instruction on **many data items** (GPU).
- **MISD** — rare/theoretical.
- **MIMD** — many processors, different instructions and data (multi-core, clusters).

A **virtual machine** is **software** that behaves like a separate computer, running its own OS on top of a host — used for isolation, testing and running several OSes at once.

## 2 Practice

Now apply the methods above.

**2.1** State **one** difference between a **RISC** and a **CISC** processor. [1]

**2.2** State what **pipelining** does. [1]

**2.3** State what **SIMD** stands for and give **one** example of SIMD hardware. [2]

**2.4** State what a **virtual machine** is. [1]

## 3 Exam-style questions

**3.1** Describe **two** features of a RISC processor that make it well-suited to **pipelining**. [2]

**3.2** Explain how **pipelining** improves performance, and name **one** type of hazard that can stall it. [3]

**3.3** State the **four** categories of **Flynn's taxonomy**. [2]

**3.4** State **one** characteristic of a **massively parallel** computer. [1]

**3.5** Identify **four** features of a **RISC** processor. [4]

## Solutions

**2.1** Any one: RISC has **few simple fixed-length** instructions, CISC has **many complex variable-length** ones; in RISC only load/store access memory.

**2.2** It **overlaps the stages** of consecutive instructions so more than one is processed at once (one finishes per cycle when full).

**2.3 Single Instruction, Multiple Data**; example: a **GPU** / graphics card / CPU vector unit.

**2.4 Software** that behaves like a separate (virtual) computer, running its own operating system on a host machine.

**3.1** Any two: **fixed-length** instructions (each stage takes equal time); **simple** instructions (each ~1 cycle); register-to-register operations (only load/store touch memory), so stages are predictable.

**3.2** While one instruction is being executed, the **next is being decoded and another fetched**, so instructions overlap and throughput rises to ~one per cycle; a **data** hazard (or control/branch hazard) can stall it.

**3.3 SISD, SIMD, MISD, MIMD**.

**3.4** Any one: **thousands of processors**; each with its **own (distributed) memory**; they communicate by **message passing**; it is MIMD.

**3.5** Any **four**: a small set of **simple** instructions; **fixed-length** instructions; only **load/store** instructions access memory; **many general-purpose registers**; a **hard-wired** control unit; designed for **pipelining** (about one instruction per cycle).

**2** Reduced Instruction Set Computers (RISC) is a type of processor.

Identify **four** features of a RISC processor.

1 .....

.....

2 .....

.....

3 .....

.....

4 .....

.....

[4]



**8** Complex Instruction Set Computer (CISC) is a type of processor.

Identify **four** features of a CISC processor.

1 .....

.....

2 .....

.....

3 .....

.....

4 .....

.....

[4]

A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

# 15.2 Boolean Algebra and Logic Circuits

Name: _____ Class: _____ Date: _____ **Total: 31 marks**

**KEY WORDS** boolean 布尔 • karnaugh map 卡诺图 • truth table 真值表 • boolean algebra 布尔代数 • flip-flop 触发器

## Objective

Build the skills to answer exam questions on **Boolean algebra and logic circuits** — simplification, Karnaugh maps, adders and flip-flops.

**You must be able to:**

- simplify **Boolean** expressions (De Morgan, absorption)
- use a **Karnaugh map** to simplify a truth table
- give the truth tables of a **half/full adder** and describe **SR/JK flip-flops**

## 1 Worked examples

Study these first. Each one shows the method for a question type used later — follow the steps and you can do the Practice and Exam-style questions yourself.

### ■ Boolean algebra

+ = OR, · = AND, overbar = NOT. Useful laws:

- **De Morgan:** $\overline{A+B} = \overline{A} \cdot \overline{B}$ and $\overline{A \cdot B} = \overline{A} + \overline{B}$
- **absorption:** $A + AB = A$
- $A + \overline{A} = 1$, so $Z = AB + A\overline{B} = A(B + \overline{B}) = A$.

Fewer terms → **fewer gates**.

### ■ Sum-of-products from a truth table

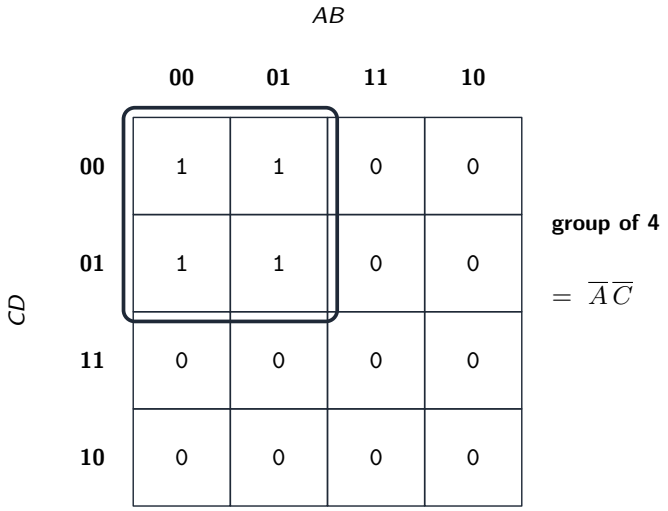
For **each row** where the output is 1, AND the inputs (a variable as itself if it is 1, or NOT it if 0); then OR these product terms together. If the 1s are at rows $A\overline{B}C$ and $ABC$:

$$Z = A\overline{B}C + ABC = AC(\overline{B} + B) = AC.$$

This is the **sum-of-products (SOP)** form; simplify it with Boolean algebra or a Karnaugh map.

### Karnaugh map

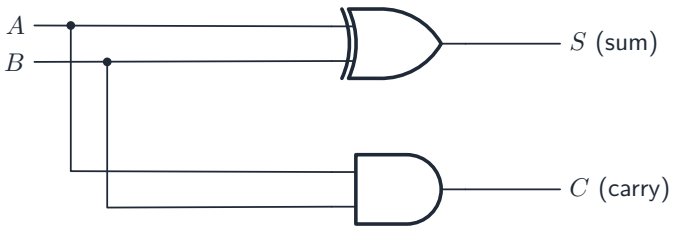
Place each truth-table `1` in the map (Gray-code order), then **group adjacent 1s** in blocks of 1, 2, 4 or 8. A larger group = a simpler term — variables that **change** within the group disappear:




A group of 2 drops 1 variable; a group of 4 drops **2**; a group of 8 drops 3.

### Half adder and flip-flops

A **half adder** gives $S = A \oplus B$ and $C = A \cdot B$:




A **full adder** also adds a carry-in. A **flip-flop** stores **one bit**: an **SR** flip-flop sets/resets/holds (`S=1,R=1` is invalid); a **JK** uses `1,1` to **toggle**, so it suits counters.

## 2 Practice

Now apply the methods above.

**2.1** Simplify $A + AB$. [1]

**2.2** Use **De Morgan's law** to rewrite $\overline{A \cdot B}$. [1]

**2.3** State the **sum** and **carry** outputs of a half adder as Boolean expressions. [2]

**2.4** State what a **flip-flop** stores. [1]

## 3 Exam-style questions

**3.1** Simplify $Z = AB + A\overline{B}$ using Boolean algebra. Show your steps. [2]

**3.2** A K-map has four 1s forming a group where $A = 0$ and $C = 0$ throughout. Write the simplified term, and state how many variables a **group of 4** removes. [2]

**3.3** Complete the truth table for a **half adder**. [3]

| A | B | S | C |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

**3.4** State **one** difference between an **SR** flip-flop and a **JK** flip-flop. [2]

**3.5** A two-input logic circuit gives output `1` only on the rows `A=0, B=1` and `A=1, B=1`. Write the Boolean expression as a **sum-of-products**, then **simplify** it. [3]

**3.6** A logic circuit has three inputs $A$, $B$ and $C$, and one output $X$, given by

$$X = \bar{A} \cdot B + A \cdot \bar{B} \cdot C + A \cdot B \cdot C$$

(a) **Complete** the truth table. [4]

| $A$ | $B$ | $C$ | $X$ |
|---|---|---|---|
| 0 | 0 | 0 | _____ |
| 0 | 0 | 1 | _____ |
| 0 | 1 | 0 | _____ |
| 0 | 1 | 1 | _____ |
| 1 | 0 | 0 | _____ |
| 1 | 0 | 1 | _____ |
| 1 | 1 | 0 | _____ |
| 1 | 1 | 1 | _____ |

(b) **Complete** the Karnaugh map for $X$. [2]

| $AB\backslash C$ | **0** | **1** |
|---|---|---|
| 00 | _____ | _____ |
| 01 | _____ | _____ |
| 11 | _____ | _____ |
| 10 | _____ | _____ |

(c) **Draw** loops around the optimal groups on your map, and **write** the simplified sum-of-products expression. [4]

(d) **Explain** why the row order of the map is 00, 01, 11, 10 and not 00, 01, 10, 11. [2]

(e) **State** how many two-input gates the simplified expression needs, compared with the original. [2]

## Solutions

**2.1** $A + AB = A$ (absorption law).

**2.2** $\overline{A \cdot B} = \overline{A} + \overline{B}$.

**2.3** $S = A \oplus B$ (A XOR B); $C = A \cdot B$ (A AND B).

**2.4 One bit** (it is bistable — it remembers its state).

**3.1** $Z = AB + A\overline{B} = A(B + \overline{B}) = A \cdot 1 = A$.

**3.2** Term $= \overline{A}\,\overline{C}$; a group of 4 removes **2** variables.

**3.3**

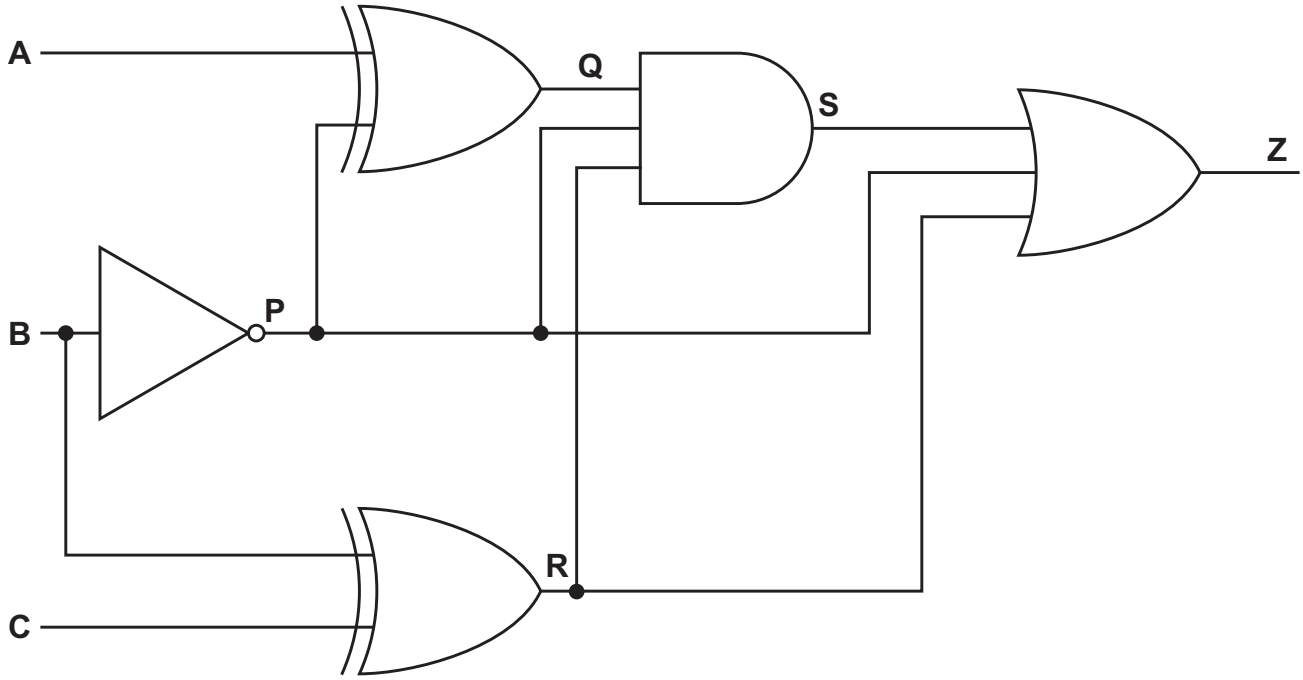
| A | B | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

**3.4** In an SR flip-flop the input `S=1, R=1` is **invalid**; a JK flip-flop uses `J=1, K=1` to **toggle** (flip) the output — so JK suits counters.

**3.5** SOP: $Z = \overline{A} \cdot B + A \cdot B$; $= B(\overline{A} + A) = B \cdot 1 = B$.

**3.6** (a) $X = 0, 0, 1, 1, 0, 1, 0, 1$ for the eight rows in the order given.
(b) Reading the same values into the map: row 00 gives $0, 0$; row 01 gives $1, 1$; row 11 gives $0, 1$; row 10 gives $0, 1$.
(c) Two groups: the whole $AB = 01$ row, which is $\bar{A}B$; and the vertical pair in the $C = 1$ column covering $AB = 11$ and $AB = 10$, which is $AC$. So $X = \bar{A}B + AC$.
(d) Consecutive rows must differ in **exactly one variable** — that is Gray-code order; it is what makes an adjacent pair of 1s correspond to a term with one variable eliminated, so grouping works at all. With 01 next to 10 two variables change and the grouping would be wrong.
(e) Simplified: one NOT, two ANDs and one OR — **four** gates. The original has three AND terms, two of them three-input, plus two NOTs and two ORs: at least **eight** two-input gates.

**6** **(a)** The diagram shows a logic circuit.



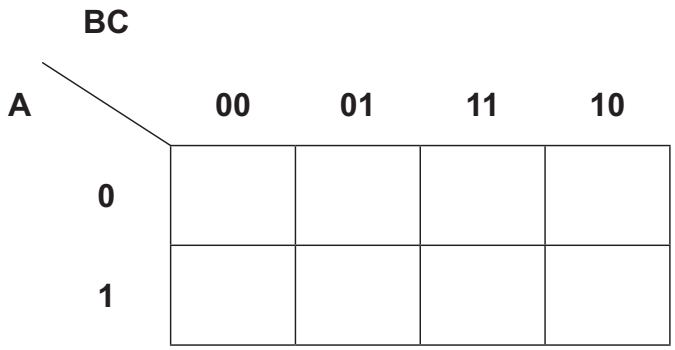

Complete the truth table for the given logic circuit.
Show your working.

| | | | Working space | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **B** | **C** | **P** | **Q** | **R** | **S** | **Z** |
| 0 | 0 | 0 | | | | | |
| 0 | 0 | 1 | | | | | |
| 0 | 1 | 0 | | | | | |
| 0 | 1 | 1 | | | | | |
| 1 | 0 | 0 | | | | | |
| 1 | 0 | 1 | | | | | |
| 1 | 1 | 0 | | | | | |
| 1 | 1 | 1 | | | | | |

[3]

**(b) (i)** Complete the Karnaugh map (K-map) for the Boolean expression:

$\overline{A}.\overline{B}.C + \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C}$




[2]

**(ii)** Draw loop(s) around appropriate group(s) in the K-map to produce an optimal sum-of-products. [2]

**(iii)** Write the Boolean expression from your answer to part **b(ii)** as a simplified sum-of-products. Do **not** carry out any further simplification.

.....

..... [2]

**6** **(a)** The Karnaugh map (K-map) represents a logic circuit with four inputs.

| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **00** | 0 | 0 | 0 | 0 |
| **01** | 1 | 1 | 1 | 0 |
| **11** | 0 | 1 | 1 | 1 |
| **10** | 0 | 0 | 1 | 1 |

**(i)** Draw loop(s) around appropriate group(s) in the K-map to produce an optimal sum-of-products. [3]

**(ii)** Write the Boolean expression from your answer to part **a(i)** as a simplified sum-of-products. Do **not** carry out any further simplification.

.....

..... [2]

**(b)** Simplify the following expression using De Morgan's laws and Boolean algebra.

Show all the stages in your simplification.

$\mathbf{X} = \overline{\mathbf{A}+\mathbf{B}+\mathbf{C}}+\overline{\overline{\mathbf{B}}+\mathbf{C}}$

.....

.....

.....

.....

.....

..... [3]

**6** The truth table for a logic circuit is shown.

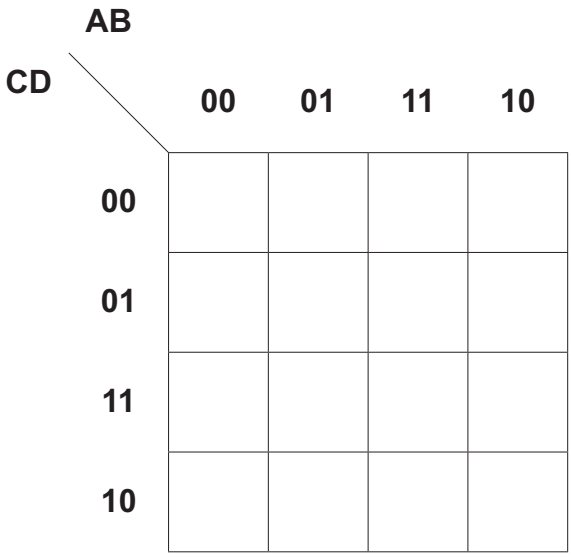
| INPUT | | | | OUTPUT |
|---|---|---|---|---|
| **A** | **B** | **C** | **D** | **X** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

**(a)** Write the Boolean logic expression that corresponds to the given truth table as the sum-of-products.

**X** = .....

..... [3]

**(b)** Complete the Karnaugh map (K-map) for the given truth table.




[2]

**(c)** Draw loop(s) around appropriate group(s) in the K-map to produce an optimal sum-of-products. [2]

**(d)** Write the Boolean logic expression from your answer to part **(c)** as the simplified sum-of-products.

**X** = .....

..... [2]

**7** The truth table for a logic circuit is shown.

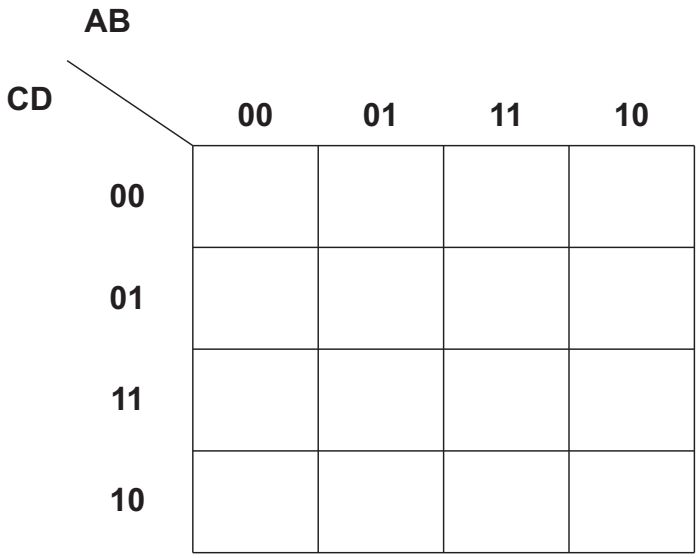
| INPUT | | | | OUTPUT |
|---|---|---|---|---|
| **A** | **B** | **C** | **D** | **T** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**(a)** Write the Boolean logic expression that corresponds to the given truth table as the sum-of-products.

**T =** .....

..... [3]

**(b)** Complete the Karnaugh map (K-map) for the given truth table.




[2]

**(c)** Draw loop(s) around appropriate group(s) in the K-map to produce an optimal sum-of-products. [2]

**(d)** **(i)** Write the Boolean logic expression from your answer to part **(c)** as the simplified sum-of-products.

**T =** .....

..... [2]

**(ii)** Use Boolean algebra to write your answer to part **(d)(i)** in its simplest form.

**T =** ..... [1]