PrepWithTee

NOTES

Software

Cambridge IGCSE Computer Science (0478)

The full notes for this chapter: the topic taught from the basics, then how it is examined - question types, marking points, keywords and real past-paper questions solved - with practice to finish.

Contents

1	System and application software	7 min
2	Firmware and interrupts	9 min
3	Programming languages and translators	10 min
4	Integrated development environments (IDEs)	6 min

Each note has two parts. **Part 1** teaches the topic from the basics. **Part 2** shows how it is examined: the types of question, what the examiner gives marks for, the keywords to use, and real past-paper questions solved with the official mark scheme. The answers to every practice question are at the end of each note.

NOTE 1

System and application software

1 In this note

SYLLABUS 4.1 TYPES OF SOFTWARE AND INTERRUPTS

- 4.1.1 Describe the difference between **system software** and **application software** and give examples of each.
- 4.1.2 Describe the role and basic **functions of an operating system**.

Next: Firmware and interrupts.

PART 1

Learn the topic

The notes: every idea from the start, with our own examples.

2 What you should already know

Hardware vs software

Hardware is the physical parts you can touch. **Software** is the programs: the instructions that tell the hardware what to do.

Quick check: is a printer driver hardware or software?

Software.

3 System software and application software

Software means the **programs / instructions** used to operate a computer. There are two types.

	System software	Application software
purpose	provides the services that **the computer requires**: it **manages and maintains** the hardware and software	provides the services that **the user requires**: it lets the user **perform tasks**
examples	**operating system**; **utility software** such as anti-virus, firewall, defragmentation, backup, compression	word processor, spreadsheet, database, web browser, photo / video editor, email client, games, social media app

> A test you can use: **would the computer still need it if nobody used it for work?** Yes → system software. If it only exists to do a job for the user → application software. A web browser is an application; an anti-virus is a utility (system).

4 The operating system

The **operating system (OS)** is the system software that **manages the hardware**, performs the **basic functions** of the computer and provides a **platform for applications to run on** and an **interface** for the user. Examples: Windows, macOS, Linux, Android, iOS.

Function	What it means
Managing files	creating, naming, saving, moving, copying and deleting files and folders; organising storage
Handling interrupts	giving each interrupt a priority, queueing it and calling the interrupt service routine (next note)
Providing an interface	a GUI (windows, icons, menus, pointer) or a command-line interface so the user can communicate with the computer
Managing peripherals and drivers	communicating with input / output devices through their drivers; sending data to and receiving it from hardware; using buffers
Managing memory	allocating RAM to each process and freeing it; preventing two processes using the same memory; virtual memory
Managing multitasking	switching the CPU between several running processes so they appear to run at once
Providing a platform for running applications	letting application software run and communicate with the hardware
Providing system security	usernames and passwords, access rights, firewall, security updates
Managing user accounts	allowing several user accounts with their own files, settings and permissions

✗ COMMON MISTAKE

Not an OS function: loading the bootstrap. The bootstrap / BIOS is **firmware** that loads the operating system (next note). Learn the nine functions above by name: questions ask you to tick the odd one out.

TRY IT YOURSELF

1. Is anti-virus system or application software?
2. Which type of software manages inputs and outputs for the computer?

3 Give one function of an operating system that relates to memory.

5 Summary

- Software = programs / instructions used to operate a computer.
- System software: services the computer requires (OS, utilities). Application software: services the user requires.
- OS functions: files · interrupts · interface · peripherals and drivers · memory · multitasking · platform for apps · security · user accounts.

IF YOU SEE …

"describe the difference between system and application software"	→ computer requires … whereas … user requires · + one example of each
"identify one function of the OS and describe its purpose"	→ name it, then one sentence from the table

PART 2

How it is examined

Question types, what the examiner gives marks for, and real past-paper questions solved.

6 Ways they test it

TYPE 1 **Difference + examples**

> *"Describe the difference between system software and application software. Give an example of each" [4]*

Spot it "difference".

1 Two marks for the contrast ("whereas"), one per example.

TYPE 2 **Which is utility?**

> *"Tick one box to show which software is an example of utility software" [1]*

Spot it a list.

1. Utilities maintain the computer: anti-virus, defragmenter, backup, compression.

TYPE 3 **OS functions**

> *"Give two functions of an operating system" [2]; "identify one function and describe its purpose" [2]; "which is not a function" [1]*

Spot it "function".

1. Use the syllabus names exactly.

TYPE 4 **Purpose of the OS**

> *"Describe the purpose of the operating system" [3]*

Spot it "purpose".

1. basic functions · manages the hardware · platform for software · user interface

7 What the examiner looks for

Wording trap · Not an OS function

THE QUESTION **0478/11/O/N/24 Q7** ↗ Paper 1 · Computer systems — **1 mark**

Tick one box to show which of these is not a function of the operating system. A managing files · B providing an interface · C handling interrupts · D loading the bootstrap

✓ CREDITED

- D: loading the bootstrap

✗ NOT CREDITED

- A, B, C (all are syllabus OS functions)

WHY

The bootstrap is firmware: it runs before the operating system and loads it.

From: 0478/11/O/N/24 Q7(a)(i) mark scheme

8 Past-paper questions, solved

0478/12/M/J/23 Q10 ↗ Paper 1 · Computer systems **5 marks** ⤢ Full size

10 A user has both system software and application software installed on their computer.

(a) Describe the difference between system software and application software.

Give an example of each software in your answer.

..... [4]

(b) State which component in the computer would store both types of software when the power is turned off.

..... [1]

Swipe sideways to read the whole question

How to start (a) contrast + one example each. (b) where does software live with the power off?

▼ **Show a full-mark answer and the mark scheme** · official marking

WHAT TO WRITE

(a) System software provides the services that the computer requires, for example an operating system or anti-virus utility, whereas application software provides the services that the user requires, for example a word processor.

(b) Secondary storage (e.g. the hard disk drive or SSD).

PART	ANSWER / WORKING	MARKS	GUIDANCE
10(a)	computer requires … whereas user requires · system example (OS / utility) · application example	B4	
10(b)	Secondary storage // HDD // SSD	B1	

Total: 5 marks

3 A computer has an operating system (OS).

(a) Complete the table by writing each missing OS function name and description.

Function name	Description
managing memory	…… …… …… ……
…… ……	allows application software to run on the computer
managing peripherals	…… …… …… ……

[3]

Swipe sideways to read the whole question

How to start Two descriptions and one function name are missing.

▾ Show a full-mark answer and the mark scheme · official marking

WHAT TO WRITE

Managing memory: allocates memory to each process and stops two processes accessing the same memory. Platform for running applications (allows application software to run). Managing peripherals: transmits data to and receives data from the hardware, using buffers.

PART	ANSWER / WORKING	MARKS	GUIDANCE
3(a)	managing memory description · platform for running applications · managing peripherals description	B3	

Total: 3 marks

0478/13/O/N/23 Q1 ↗ Paper 1 · Computer systems — **3 marks** — ⤢ Full size

1 A mobile telephone has built-in input and output devices.

(d) The mobile telephone has an operating system.

Describe the purpose of the operating system.

..... [3]

Swipe sideways to read the whole question

How to start Three marks: three separate points about what the OS does.

▾ **Show a full-mark answer and the mark scheme** · official marking

WHAT TO WRITE

It manages the phone's hardware, such as the touch screen and memory. It provides a platform for apps to run on. It provides a user interface so the user can communicate with the phone.

PART	ANSWER / WORKING	MARKS	GUIDANCE
1(d)	Any three: basic functions · manages the hardware · platform to run software · user interface · any example function	B3	

Total: 3 marks

9 Common mistakes

✗ "A web browser is system software" ✓ "A web browser is application software"

✗ "The OS loads the bootstrap" ✓ "The bootstrap (firmware) loads the OS"

✗ "Memory" as an OS function ✓ "Managing memory"

Name the function, not just the hardware.

10 Practice

Practice: easy, medium, hard

1. Is a spreadsheet system or application software? [1]
2. Give two examples of utility software. [2]
3. Describe what is meant by multitasking. [1]

2 State what is meant by software.

1 A school wants each student to have their own files that others cannot open. Which two OS functions make this possible? [2]

11 Can I do this?

0 / 2 ticked

- [ ] I can describe the difference between system and application software with examples.
- [ ] I can name and describe the functions of an operating system.

Anything unticked: revise that section first (each line links to it). All ticked: take the quiz.

Answers

Answers to the "Try it yourself" and practice questions on this page. Try every question first.

Try it yourself 1

1.

ANSWER System (utility) software

2.

ANSWER Operating system (system software)

3.

ANSWER Managing memory (or managing multitasking)

Practice: easy, medium, hard

Easy 1.

ANSWER Application

Easy 2.

ANSWER e.g. anti-virus, defragmentation (also backup, compression, firewall)

Medium 1.

ANSWER

The OS switches the CPU between several processes so they appear to run at the same time

Medium 2.

ANSWER Programs / instructions used to operate a computer

Hard 1.

ANSWER Managing user accounts; providing system security (also managing files)

NOTE 2

Firmware and interrupts

1 In this note

SYLLABUS 4.1 TYPES OF SOFTWARE AND INTERRUPTS

- 4.1.3 Understand how **hardware**, **firmware** and an **operating system** are required to run **applications software**.
- 4.1.4 Describe the role and operation of **interrupts**: how one is **generated**, how it is **handled using an interrupt service routine**, and what happens as a result.

Before this: System and application software. Next: Programming languages and translators.

PART 1

Learn the topic

The notes: every idea from the start, with our own examples.

2 What you should already know

ROM

ROM is non-volatile, read-only primary storage. It keeps its contents with the power off.

Quick check: what does ROM store?

The start-up instructions: the BIOS / bootstrap (firmware).

The FDE cycle

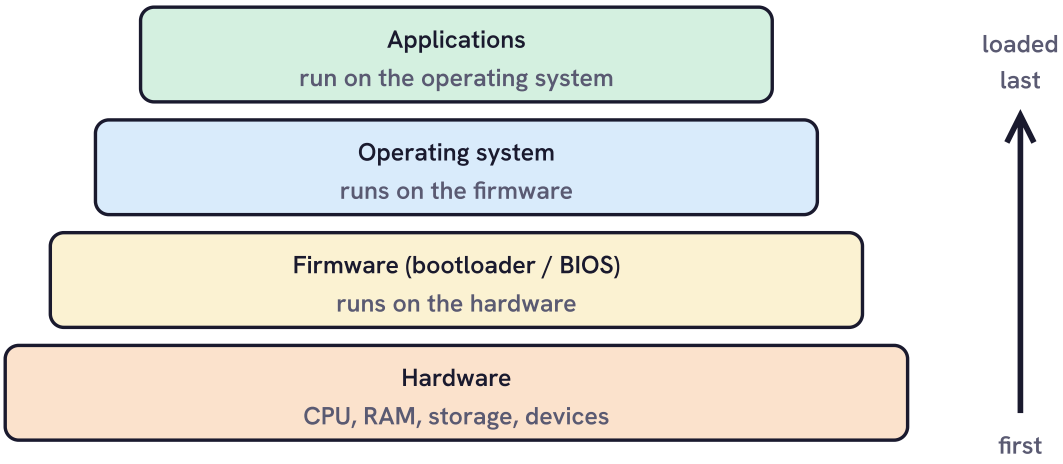
The CPU fetches, decodes and executes one instruction at a time.

Quick check: when can the CPU check for interrupts?

At the end of each fetch-decode-execute cycle.

3 Hardware, firmware, OS, applications

Firmware is software **permanently stored in ROM** that **controls the hardware**. The most important firmware is the **bootloader** (also called the bootstrap / BIOS): the instructions that **start up the computer** and load the operating system.



Each layer runs on the one below it.

GENERAL METHOD **What happens when you switch on**

1. The **hardware** powers up and the CPU runs the **bootloader**, which is **firmware stored in ROM**.
2. The bootloader **loads the operating system** from secondary storage into RAM: the OS **runs on the firmware**.
3. The OS gives a **platform** for the **applications**, which **run on the operating system**.

4 Interrupts

An **interrupt** is a **signal sent to the CPU** to show that **something needs its attention**. Interrupts let the computer react straight away to events (a key press, a printer running out of

paper) instead of checking every device all the time. They are managed by the **operating system** (its **interrupt handler**).

Hardware interrupts (from a device)	Software interrupts (from a program)
pressing a key on the keyboard; **moving the mouse** or clicking a button; plugging in a device; printer out of paper / ink; paper jam	**division by zero**; **two processes trying to access the same memory location**; a program asking for input / output; a null value error

GENERAL METHOD **How an interrupt is handled**

1. A device or program **generates an interrupt** (e.g. a key is pressed) and it is **sent to the CPU**.
2. The interrupt is given a **priority** and placed in the **interrupt queue**.
3. The CPU **finishes its current fetch-decode-execute cycle**, then **checks the interrupt queue**.
4. If the interrupt has a **higher priority** than the current task, the CPU **saves the current task** (the contents of its registers) and **calls the interrupt service routine (ISR)** for that interrupt.
5. The **ISR handles the interrupt** (e.g. reads the key, shows the "out of paper" message).
6. The CPU checks the queue again; when there are no more higher-priority interrupts it **restores the saved task** and carries on.
7. If the interrupt has a **lower priority**, the CPU simply continues the current task and the interrupt waits in the queue.

Interrupt service routine (ISR) (also called the interrupt handler): the sequence of instructions that handles one particular interrupt.

EXAMPLE **A printer runs out of paper**

Describe what happens.

The printer sends an interrupt to the CPU.

The interrupt is given a priority and placed in the interrupt queue.

The CPU finishes its current FDE cycle and checks the queue: the paper interrupt has a higher priority.

The CPU saves its current task and calls the ISR for the printer.

The ISR displays a message telling the user to add paper; the CPU then restores the saved task.

ANSWER

Result: the user sees "Out of paper" straight away, and the program they were using carries on.

TRY IT YOURSELF

1. Where is the bootloader stored?
2. Is division by zero a hardware or software interrupt?
3. What is the name of the program that services an interrupt?

5 Summary

- Applications run on the OS; the OS runs on the firmware; the bootloader (firmware, in ROM) runs on the hardware.
- Interrupt: a signal to the CPU that something needs attention; managed by the OS.
- Hardware interrupts: key press, mouse, printer. Software: division by zero, two processes same memory.
- Handling: priority → queue → CPU finishes FDE cycle → checks queue → saves task → ISR → restores task.

IF YOU SEE …

"explain how the processor manages the current program and the interrupt" [5]	→ priority · queue · finish FDE cycle · check queue · store task · ISR · restore
"state the purpose of an interrupt"	→ to indicate something needs the attention of the CPU

PART 2

How it is examined

Question types, what the examiner gives marks for, and real past-paper questions solved.

6 Ways they test it

TYPE 1 **Layers cloze**

> *"... software is run on the operating system. The ... system is run on the firmware, which is run on the ..." [4]*

Spot it a word list.

1 Applications → OS → firmware → hardware.

TYPE 2 **Firmware**

> *"State the purpose of firmware" [1]; "give one example" [1]*

Spot it "firmware".

1. Purpose: permanently stores instructions to start the computer / control the hardware. Example: bootloader, BIOS.

TYPE 3 **Examples**

> *"Give two examples of a hardware / software interrupt" [2]*

Spot it "examples".

1. Use the table.

TYPE 4 **Handling**

> *"Describe how interrupts are used when a key is pressed" [5]; the printer paragraph [7]*

Spot it "describe how".

1. Follow the handling method in order, naming the device from the question.

7 What the examiner looks for

"Describe how interrupts are used when a key is pressed on a keyboard" [5]

Marking points (one mark each, up to the marks shown)

- Key press generates the interrupt
- Interrupt is given a priority
- Interrupt is sent to the CPU … and placed in a queue
- CPU stops its current task (at the end of the FDE cycle) to check the queue / service the interrupt …
- … using an interrupt service routine
- If the key press is the highest priority the interrupt is processed

KEYWORDS **priority** **queue** **interrupt service routine**

8 Past-paper questions, solved

0478/11/M/J/23 Q4 ↗ Paper 1 · Computer systems **4 marks** ⤢ Full size

4 Complete the statements about different types of software.

Use the terms from the list.

Some of the terms in the list will **not** be used. You should only use a term once.

application assembly language bootloader central processing unit (CPU)
firmware hardware operating output system user

..... software provides the services that the computer requires; an example is utility software.

..... software is run on the operating system.

The ..... system is run on the firmware, which is run on the ..... .

[4]

Swipe sideways to read the whole question

How to start Use the layer diagram: who runs on whom?

▼ Show a full-mark answer and the mark scheme · official marking

WHAT TO WRITE

System software provides the services that the computer requires … **Application** software is run on the operating system. The **operating** system is run on the firmware, which is run on the **hardware**.

PART	ANSWER / WORKING	MARKS	GUIDANCE
4	System · Application · Operating · Hardware	B4	

Total: 4 marks

9 A printer runs out of ink and needs a new ink cartridge. An interrupt is sent and a message is generated to notify the user that the ink cartridge is empty.

(a) Complete the paragraph about the use of an interrupt in this process.

Use the terms from the list.

Some of the terms in the list will **not** be used. You should only use a term once.

binary clock computer data error

fetch–decode–execute cycle hexadecimal higher interrupt queue

interrupt service routine (ISR) lower printer priority level

secondary storage sorting transmission

The ..... sends an interrupt to the

..... . The interrupt is given a

..... . The processor completes its current

..... and checks the

..... to see if there is an interrupt with

..... priority than the current task. If there is, it stores the current task and fetches the interrupt. It then calls the

..... , which is a sequence of instructions that handles the interrupt. The interrupt is handled, which generates a message to the user to notify them that the ink cartridge is empty.

[7]

(b) Handling interrupts is one function of an operating system.

Give **two** functions of an operating system that relate to memory.

1 .....

2 .....

[2]

Swipe sideways to read the whole question

How to start (a) follow the handling method. (b) which OS functions are about memory?

▼ **Show a full-mark answer and the mark scheme** · official marking

WHAT TO WRITE

(a) printer · computer · priority level · fetch-decode-execute cycle · interrupt queue · higher · interrupt service routine (ISR)

(b) Managing memory; managing multitasking.

PART	ANSWER / WORKING	MARKS	GUIDANCE
9(a)	printer · computer · priority level · FDE cycle · interrupt queue · higher · ISR	B7	
9(b)	memory management · multitasking	B2	

Total: 9 marks

9 An interrupt is a type of signal that is used in a computer.

(a) State the name of the type of software that manages interrupts.

..... [1]

(b) Describe how interrupts are used when a key is pressed on a keyboard.

..... [5]

(c) Interrupts can be hardware based or software based.

A key press is one example of a hardware interrupt.

(i) Give **two** other examples of a hardware interrupt.

[2]

(ii) Give **two** examples of a software interrupt.

[2]

Swipe sideways to read the whole question

How to start (b) is the handling method for a key press. (c) two of each kind.

▼ **Show a full-mark answer and the mark scheme** · official marking

WHAT TO WRITE

(a) The operating system.

(b) Pressing the key generates an interrupt, which is sent to the CPU. It is given a priority and placed in the interrupt queue. The CPU finishes its current FDE cycle and checks the queue. If the key press has the highest priority, the CPU saves its current task and services the interrupt using an interrupt service routine.

(c)(i) Moving the mouse; a printer running out of paper. (ii) Division by zero; two processes accessing the same memory location.

PART	ANSWER / WORKING	MARKS	GUIDANCE
9(a)	Operating system // interrupt handler	**B1**	
9(b)	Any five: key press generates · priority · sent to CPU · queue · CPU stops current task / checks queue · ISR · highest priority processed	**B5**	
9(c)(i)	Any two: moving / clicking the mouse · plugging in a device · paper jam · out of paper	**B2**	
9(c)(ii)	Any two: division by zero · two processes same memory location · null value	**B2**	

Total: 10 marks

9 Common mistakes

✗ "The CPU stops immediately"

✓ "The CPU finishes its current FDE cycle first"

✗	✓
"The interrupt is always handled straight away"	"Only if it has a higher priority than the current task"
"Firmware is stored in RAM"	"Firmware is permanently stored in ROM"
"Typing a letter" as a software interrupt	A key press is a hardware interrupt

10 Practice

Practice: easy, medium, hard

1. Which software runs directly on the hardware?
2. Give one example of a hardware interrupt.
1. State the purpose of an interrupt.
2. Explain why the CPU saves its current task before servicing an interrupt.
1. A program divides a number by zero while a user is moving the mouse. Explain how the CPU decides which interrupt to service first. [2]

11 Can I do this?

0 / 3 ticked

- [ ] I can explain how hardware, firmware and the OS are needed to run applications.
- [ ] I can give examples of hardware and software interrupts.
- [ ] I can describe how an interrupt is handled using an ISR.

Anything unticked: revise that section first (each line links to it). All ticked: take the quiz.

Answers

Answers to the "Try it yourself" and practice questions on this page. Try every question first.

Try it yourself 1

1.

ANSWER In ROM (it is firmware)

2.

ANSWER Software

3.

ANSWER Interrupt service routine (ISR) / interrupt handler

Practice: easy, medium, hard

Easy 1.

ANSWER Firmware (the bootloader)

Easy 2.

ANSWER e.g. a key press

Medium 1.

ANSWER To indicate that something needs the attention of the CPU

Medium 2.

ANSWER So it can restore it and carry on from the same point afterwards

Hard 1.

ANSWER

Each interrupt is given a priority and placed in a queue; the CPU services the highest priority one first

NOTE 3

Programming languages and translators

1 In this note

SYLLABUS 4.2 TYPES OF PROGRAMMING LANGUAGE, TRANSLATORS AND IDES

- 4.2.1 Explain what is meant by a **high-level language** and a **low-level language**, including the advantages and disadvantages of each.
- 4.2.2 Understand that **assembly language** is a low-level language that uses **mnemonics**, and that an **assembler** translates it into machine code.
- 4.2.3 Describe the operation of a **compiler** and an **interpreter**: how each translates and how **errors are reported**.
- 4.2.4 Explain the advantages and disadvantages of a compiler and an interpreter.

Before this: Firmware and interrupts. Next: Integrated development environments (IDEs).

PART 1

Learn the topic

The notes: every idea from the start, with our own examples.

2 What you should already know

Machine code

The CPU can only execute **machine code**: binary instructions from its instruction set.

Quick check: can the CPU run Python directly?

No: it must be translated into machine code first.

3 High-level and low-level languages

A **high-level language** is **close to human language** (English words and maths symbols), e.g. Python, Java, Visual Basic. A **low-level language** is **close to the language processed by the computer**: **machine code** (binary) or **assembly language**.

The same job at three levels	Code
high-level	`Total ← Price + Tax`

assembly language (low-level)	`LDA Price`
	`ADD Tax`
	`STO Total`

machine code (low-level)	`0001 1010`
	`0011 1011`
	`0010 1100`

	High-level language	Low-level language
advantages	**easier to read, write and understand** → fewer mistakes, written faster **easier to debug** **machine independent** (portable): runs on different types of computer programmer can focus on the problem, not the hardware	**direct manipulation of the hardware** / memory **more memory efficient**, **faster to execute** can use specialised / machine-specific instructions (machine code) needs no translator
disadvantages	**cannot directly manipulate the hardware** may be less efficient (uses more memory, slower) needs a compiler or interpreter	**hard to read and write** **hard to debug** **machine dependent**: only works on one type of CPU needs knowledge of the hardware

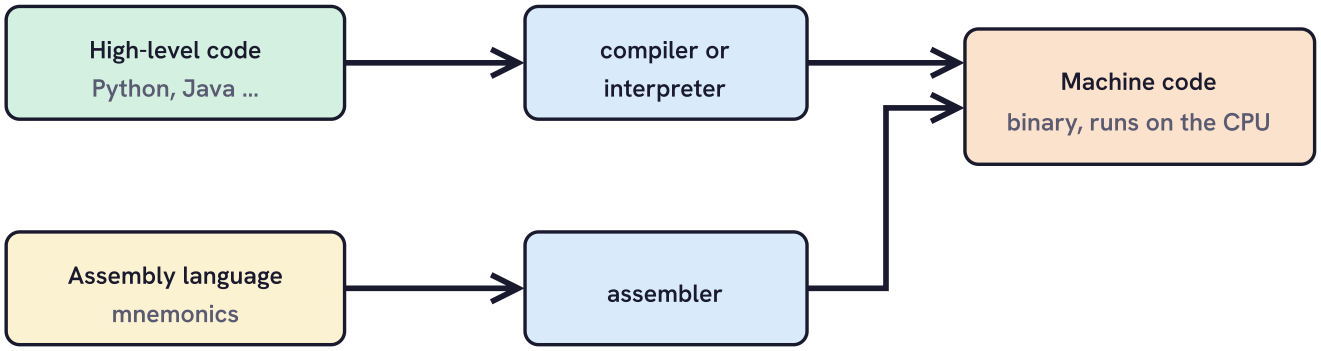
> Choosing: a program that must be written **quickly**, be **easy to maintain** or run on **many computers** → high-level. A program that must **control hardware directly**, use **little memory** or run **very fast** (e.g. an embedded system in a vending machine) → low-level.

4 Assembly language and the assembler

Assembly language is a **low-level language** that uses **mnemonics**: short codes such as LDA (load), ADD, STO (store) that stand for machine code instructions. Each assembly instruction

usually matches one machine code instruction. An **assembler** is the translator that converts assembly language into machine code.

5 Compilers and interpreters



Every program must become machine code before the CPU can execute it.

	Compiler	Interpreter
how it translates	translates the **whole code at once before executing** it	translates and executes the code **line by line**
output	produces an **executable file** (object code) that runs without the compiler	**no executable file**: the interpreter is needed every time it runs
errors	produces an **error report for the whole code**, listing all errors; nothing runs until they are fixed	**stops execution when an error is found**; once it is corrected, execution can continue from that point
speed of running	the executable runs **fast** (already translated)	**slower**: every line is translated each time it runs
best used	to translate the **final program** for distribution (the source code is not given away)	**while developing** a program: test and debug parts as you go

Advantages of an interpreter during development	Advantages of a compiler for the final program
errors are found **line by line**, where they happen → easy to debug can **correct and continue** without retranslating everything can **test sections** of incomplete code	executable file **runs without the translator** **runs faster** users **cannot see the source code** (harder to copy) all errors reported at once

TRY IT YOURSELF

1. Which translator converts assembly language into machine code?
2. Which translator produces an executable file?
3. Which translator stops when it finds an error?
4. Is a high-level language machine dependent or independent?

6 Summary

- High-level: close to English; easy to read, write, debug; machine independent; can't manipulate hardware directly.
- Low-level (machine code, assembly): close to the computer; direct hardware control, memory efficient, fast; hard to read/debug, machine dependent.
- Assembly language uses mnemonics; an assembler translates it.
- Compiler: whole code at once, executable file, error report of all errors; used for the final program.
- Interpreter: line by line, stops at an error, no executable; used during development.

IF YOU SEE …

"describe the operation of a compiler and of an interpreter"	→ whole code / executable / error report vs line by line / stops at error
"explain the reasons for your choice" (high or low)	→ match each reason to the scenario: "… so the game can run on any computer"

PART 2

How it is examined

Question types, what the examiner gives marks for, and real past-paper questions solved.

7 Ways they test it

TYPE 1 Choose a level

> *"Tick high-level or low-level. Explain the reasons for your choice" [4]*

Spot it a scenario.

1. No mark for the tick: all four marks are for reasons that match it.

TYPE 2 Low-level / assembly

> *"Describe what is meant by assembly language" [2]; "identify the translator" [1]*

Spot it "assembly".

1. low-level · mnemonics · communicates directly with hardware · assembler

TYPE 3 Which translator?

> *"Identify the translator that converts and executes line by line / creates an executable" [1] each*

Spot it a description.

1. line by line / stops → interpreter; whole code / executable / error report → compiler; mnemonics → assembler.

TYPE 4 Translator cloze

> *"A compiler translates the … at once before … it …" [5]*

Spot it a word list.

1. Use the compiler / interpreter table.

TYPE 5 **Advantages**

> *"Describe the advantages of using an interpreter instead of a compiler during software development" [4]*

Spot it "during development".

1 Line by line errors · correct and continue · no retranslating · test sections.

8 What the examiner looks for

"Describe how the compiler translates the program" [3] and "how it reports errors" [2]

Marking points (one mark each, up to the marks shown)

- It translates the high-level language into low-level language / machine code / object code
- It translates all the code before it is executed
- It creates an executable file
- It creates an error report after trying to compile … displaying all the errors in the code
- … which must be corrected before the code can be executed

KEYWORDS **all the code** **before executed** **executable file** **error report**

9 Past-paper questions, solved

0478/12/F/M/23 Q2 ↗ Paper 1 · Computer systems **8 marks** ⤢ Full size

2 A programmer has designed a three-dimensional (3D) interactive computer game. They are going to develop a program for the game. The program needs to run efficiently, but it must also be developed as soon as possible.

(a) Tick (✓) **one** box to identify whether the programmer should use a high-level language or a low-level language to develop the program.

Explain the reasons for your choice.

High-level language ☐

Low-level language ☐

Reasons for your choice .....

.....

.....

.....

.....

.....

.....

..... [4]

(b) If the programmer chooses a high-level language, they can use a compiler or an interpreter to translate the high-level language into a low-level language.

Describe the operation of a compiler and of an interpreter.

Compiler .....

.....

.....

.....

Interpreter .....

.....

.....

.....

[4]

Swipe sideways to read the whole question

How to start The scenario says "developed as soon as possible", which points to high-level. Then give four reasons with their effect.

▼ Show a full-mark answer and the mark scheme · official marking

WHAT TO WRITE

(a) High-level language. It is easier for the programmer to read and write, so they make fewer mistakes and can write it in a shorter time. It is easier to debug, so errors are found and corrected faster. It is machine independent, so the game can be played on many different computers.

(b) Compiler: translates all the code before executing any of it, produces an executable file, and gives an error report listing all the errors. Interpreter: translates and executes one line at a time, and stops when it finds an error.

PART	ANSWER / WORKING	MARKS	GUIDANCE
2(a)	No mark for choice. Any four matching reasons: easier to read / write … fewer mistakes / quicker · easier to debug … less time · machine independent … any computer (or low-level: memory efficient · direct manipulation …)	B4	
2(b)	Compiler, two from: checks all code before executing · error report for whole code · executable file. Interpreter, two from: one line translated and executed at a time · stops at an error	B4	

Total: 8 marks

3 A programmer writes a program to store and sort all the data for the books in her home library.

(b) The programmer uses an integrated development environment (IDE) to write the program.

One function of the IDE is to provide a translator.

(i) Complete the statements about translators.

Use the terms from the list. Some of the terms will **not** be used. You should only use a term once.

all together	all	assembler	binary	error
executable file	executing	interrupt	line by line	
partial code	some	translating	whole code	

A compiler translates the ..... at once before

..... it. A compiler produces an error report that displays

..... errors.

An interpreter translates and executes the code ..... .

An interpreter stops execution when an ..... is found

and continues once it is corrected.

[5]

Swipe sideways to read the whole question

How to start Five gaps, all from the compiler / interpreter table.

▾ Show a full-mark answer and the mark scheme · official marking

WHAT TO WRITE

A compiler translates the **whole code** at once before **executing** it. A compiler produces an error report that displays **all** errors. An interpreter translates and executes the code **line by line**. An interpreter stops execution when an **error** is found and continues once it is corrected.

PART	ANSWER / WORKING	MARKS	GUIDANCE
3(b)(i)	whole code · executing · all · line by line · error	**B5**	

Total: 5 marks

0478/12/O/N/25 Q3 ↗ Paper 1 · Computer systems **4 marks** ⤢ Full size

3 A programmer uses assembly language to create a program for a coffee machine.

(a) Describe what is meant by assembly language.

..... [2]

(b) Identify the translator needed for assembly language.

..... [1]

(c) Tick (✓) **one** box to show which statement is an advantage of using assembly language.

A It is close to human language. ☐

B It is memory efficient. ☐

C It is easy to debug. ☐

D It is machine independent. ☐

[1]

Swipe sideways to read the whole question

How to start (a) two facts about assembly. (c) which option is a LOW-level advantage?

▼ **Show a full-mark answer and the mark scheme** · official marking

WHAT TO WRITE

(a) It is a low-level language that uses mnemonics.

(b) Assembler.

(c) B: it is memory efficient.

PART	ANSWER / WORKING	MARKS	GUIDANCE
3(a)	Any two: low-level · mnemonics · communicates directly with the hardware	B2	
3(b)	Assembler	B1	
3(c)	B	B1	

Total: 4 marks

10 Common mistakes

✗ "A compiler translates line by line" ✓ "An interpreter translates line by line"

✗ "An assembler translates high-level code" ✓ "An assembler translates assembly language"

✗ "High-level is easier" (no reason) ✓ "easier to read and write, so fewer mistakes"

Every reason needs its effect.

✗ "Low-level languages are machine independent" ✓ "High-level languages are machine independent"

11 Practice

Practice: easy, medium, hard

1. Give one example of a low-level language.
2. What are the short codes in assembly language called?
1. Give two reasons a programmer might write a vending machine program in a low-level language.
2. Give two disadvantages of a high-level language.
1. A company sells a finished game. Explain why it is compiled rather than interpreted. [3]

12 Can I do this?

0 / 4 ticked

- [ ] I can explain high- and low-level languages with advantages and disadvantages.
- [ ] I can describe assembly language and the assembler.
- [ ] I can describe how a compiler and an interpreter translate and report errors.
- [ ] I can explain the advantages of each translator.

Anything unticked: revise that section first (each line links to it). All ticked: take the quiz.

Answers

Answers to the "Try it yourself" and practice questions on this page. Try every question first.

Try it yourself 1

1.

ANSWER **Assembler**

2.

ANSWER **Compiler**

3.

ANSWER **Interpreter**

4.

ANSWER **Independent**

Practice: easy, medium, hard

Easy 1.

ANSWER **Assembly language (or machine code)**

Easy 2.

ANSWER **Mnemonics**

Medium 1.

ANSWER

Direct manipulation of hardware; more memory efficient (also faster, specialised hardware)

Medium 2.

ANSWER

Cannot directly manipulate hardware; needs a translator (also less memory efficient)

Hard 1.

ANSWER

Executable runs without the translator; runs faster; users cannot see / copy the source code

NOTE 4

Integrated development environments (IDEs)

1 In this note

SYLLABUS 4.2 TYPES OF PROGRAMMING LANGUAGE, TRANSLATORS AND IDES

- 4.2.5 Explain the role of an **IDE** in writing program code and the common functions IDEs provide: **code editors**, **run-time environment**, **translators**, **error diagnostics**, **auto-completion**, **auto-correction**, **prettyprint**.

Before this: Programming languages and translators.

PART 1

Learn the topic

The notes: every idea from the start, with our own examples.

2 What you should already know

Translators

A compiler or interpreter turns high-level code into machine code.

Quick check: which translator is handy while you are still testing?

An interpreter.

3 What an IDE is

An **integrated development environment (IDE)** is a **piece of software** that gives a programmer, in one place, the tools needed to **write, edit, translate, test and debug** program code.

Examples: Visual Studio Code, IDLE (Python), PyCharm, Visual Studio. Its role is to make writing correct code **quicker and easier**.

Function	What it does for the programmer
Code editor	lets the programmer **type and amend** the program code
Run-time environment	lets the programmer **run the program** inside the IDE and **see the output**
Translator	a built-in **compiler and / or interpreter** converts the code into machine code
Error diagnostics	**shows where errors are** in the code (underlining, line numbers, messages) so they can be found and fixed
Auto-completion	as the programmer starts typing a command or identifier, the IDE **suggests options to complete it**
Auto-correction	**corrects** a command word with a **minor misspelling** automatically
Prettyprint	**colours key words** and **indents** the code so its structure is easy to read (syntax highlighting)

> In an "identify and describe" question, each function gets **one mark for the name and one for its role**. "It has a compiler" is often given in the question: choose the other functions.

TRY IT YOURSELF

1. Which IDE function suggests how to finish a command as you type?
2. Which IDE function colours the key words?
3. Which IDE function lets you run the program and see the output?

4 Summary

- IDE: software with the tools to write, translate, test and debug code in one place.
- Functions: code editor · run-time environment · translator · error diagnostics · auto-completion · auto-correction · prettyprint.

IF YOU SEE …

"explain the purpose of the IDE" [4]	→ software · write / edit code · test / debug · features like auto-completion · translate
"give two functions and state the role of each" [4]	→ name + what it does, from the table

PART 2

How it is examined

Question types, what the examiner gives marks for, and real past-paper questions solved.

5 Ways they test it

TYPE 1 Name functions

> *"Give three other common functions of an IDE" [3]*

Spot it "functions".

1. Names from the syllabus list.

TYPE 2 Function + role

> *"Describe three other ways the IDE can help the programmer" [6]*

Spot it "describe", "help".

1. Name + what it does for the programmer, for each.

TYPE 3 **Purpose**

> *"Explain the purpose of the IDE" [4]*

Spot it "purpose".

1 Software · write · test / debug · translate · example feature.

6 What the examiner looks for

"Describe three other ways the IDE can help the programmer" [6]

Marking points (one mark each, up to the marks shown)

- Code editor … allows the user to enter and amend code
- Run-time environment … allows the program to be run and its outputs seen
- Error diagnostics … shows the programmer where there are errors
- Auto-completion … gives options / suggestions of key commands
- Auto-correction … corrects a command with a minor misspelling
- Prettyprint … colours the key commands so they are easy to identify

KEYWORDS **code editor** **run-time environment** **error diagnostics** **auto-completion** **auto-correction** **prettyprint**

TOO VAGUE "it helps you code" (no named function); the translator, when the question already gave it

7 Past-paper questions, solved

0478/11/O/N/24 Q6 ↗ Paper 1 · Computer systems **6 marks** ⤢ Full size

6 A computer programmer uses assembly language to create a computer program for an embedded system in a washing machine.

(e) The programmer uses an integrated development environment (IDE) to write the program.

A built-in translator to convert the program into machine code is one way that the IDE helps the computer programmer.

Describe **three** other ways the IDE can help the programmer.

..... [6]

Swipe sideways to read the whole question

How to start The question gave the translator, so choose three other functions: name + role each.

▼ **Show a full-mark answer and the mark scheme** · official marking

WHAT TO WRITE

Error diagnostics show the programmer where the errors are in the program so they can correct them. Auto-completion gives the programmer suggestions to complete a command as they type it. Prettyprint colours the key commands so they are easy to identify.

PART	ANSWER / WORKING	MARKS	GUIDANCE
6(e)	Six marks, max 4 for naming: code editor · run-time environment · error diagnostics · auto-completion · auto-correction · prettyprint, each with its role	B6	

Total: 6 marks

0478/13/O/N/24 Q7 ↗ Paper 1 · Computer systems **4 marks** ⤢ Full size

7 A computer programmer uses an integrated development environment (IDE) when creating a computer program.

Explain the purpose of the IDE.

..... [4]

Swipe sideways to read the whole question

How to start Four separate points about what an IDE is and does.

▾ **Show a full-mark answer and the mark scheme** · official marking

WHAT TO WRITE

An IDE is a piece of software used to write and edit program code. It is used to test and debug the code, with features such as error diagnostics and auto-completion. It also translates the code into machine code.

PART	ANSWER / WORKING	MARKS	GUIDANCE
7	Any four: piece of software · used to write / edit code · test / debug · features such as auto-completion · translate to machine code	B4	

Total: 4 marks

8 Common mistakes

✗ "Auto-correct finds logic errors"

✓ "Auto-correction fixes minor misspellings of command words"

✗ "Prettyprint prints the code"

✓ "Prettyprint colours / indents the code to make it readable"

✗ "The IDE runs the program" as the only point

✓ name several functions with their roles

9 Practice

Practice: easy, medium, hard

1 What does IDE stand for?

2 Name the IDE function that lets the programmer type and amend code.

1 Describe error diagnostics. [2]

1 A beginner keeps misspelling PRINT and forgetting the end of long variable names. Which two IDE functions help, and how? [4]

10 Can I do this?

0 / 2 ticked

- [ ] I can explain the purpose of an IDE.
- [ ] I can name and describe the seven common IDE functions.

Anything unticked: revise that section first (each line links to it). All ticked: take the quiz.

Answers

Answers to the "Try it yourself" and practice questions on this page. Try every question first.

Try it yourself 1

1.

ANSWER **Auto-completion**

2.

ANSWER **Prettyprint**

3.

ANSWER **Run-time environment**

Practice: easy, medium, hard

Easy 1.

ANSWER **Integrated development environment**

Easy 2.

ANSWER **Code editor**

Medium 1.

ANSWER

An IDE function that shows where errors are in the code, e.g. underlines / gives the line number, so they can be fixed

Hard 1.

ANSWER

Auto-correction corrects the misspelt command; auto-completion suggests the full variable name as they type