CHAPTER 4 – FULL CHAPTER WORKSHEET

ANSWERS

4.1 Types of software and interrupts

0478/13 – JUNE 2019

Question	Answer	Marks
2(c)(i)	– Interrupt	1
2(c)(ii)	**Two** from: – Provides an interface – Loads / opens / installs / closes software – Manages the hardware // manages peripherals // spooling – Manages the transfer of programs into and out of memory – Divides processing time // processor management – Manages file handling – Manages error handling // interrupt handling – Manages security software – Manages utility software – Manages user accounts – Multitasking – Multiprogramming // time slicing – Batch processing	2

0478/13 – NOVEMBER 2019

(b) **One** mark for each correct row

Statement	True (✓)	False (✓)
Interrupts can be hardware based or software based	✓	
Interrupts are handled by the operating system	✓	
Interrupts allow a computer to multitask	✓	
Interrupts work out which program to give priority to		✓
Interrupts are vital to a computer and it cannot function without them	✓	

0478/12 – JUNE 2020

(c)(i)	Any **two** from: – Paper jam – Out of paper – Out of toner/ink – Buffer full – Awaiting input – Print complete – Printer ready Award any other valid example
(c)(ii)	Any **one** from: – Operating system – Interrupt handler – Interrupt service routine

0478/11 – MARCH 2021

(c)	Any **two** from: – To identify that the processor's attention is required // to stop the current **process/task** – To allow multitasking – To allow for efficient processing // prioritising actions – To allow for efficient use of hardware – To allow time-sensitive requests to be dealt with – To avoid the need to poll devices

NOVEMBER 2021 (2)

9(a)	Any **three** from: e.g. – A suitable description of any error that might occur – A peripheral is connected/disconnected – A key on a keyboard is pressed – A mouse button click – A phone/video call is received – A buffer requires more data – A printer has a paper jam – A printer runs out of paper – A printer runs out of ink – When switching from one application to another NOTE: If three suitable different errors are described, this can be awarded three marks.
9(b)	Any **one** from: – The computer would only start a new task when it had finished processing the current task // by example – Computer will not be able to multitask – Errors may not be dealt with – Computer would become impossible to use

NOVEMBER 2021 (3)

uestion	Answer
4(a)	Any **four** from: – Printer generates interrupt – Interrupt is given a priority – Interrupt is queued – Interrupt stops **CPU** from processing current task – CPU will service interrupt // Interrupt handler services interrupt … – … generating an output message to state there is a paper jam

JUNE 2023

4	**One** mark for each correct term in the correct place: • System • Application • Operating • Hardware

JUNE 2023 (2)

10(a)	**Two** from: • System software provides services that the computer requires • … whereas application software provides services that the user requires **One** from (system software): • Utility software // by example e.g. defragmentation software, antivirus, firewall • Operating system **One** from (application software): • Any suitable example of an application e.g. word processor, web browser, video-editing software
10(b)	• Secondary storage // HDD // SSD

JUNE 2023 (3)

11(a)	• Operating system
11(b)	Any **one** from: • Create a file • Copy a file • Open a file • Close a file • Move a file • Delete a file • Rename a file • Save a file • Sort files

11(c)	Any **two** from: e.g. • Keeping track of the status of each memory location • Managing the movement of data to and from RAM • Checks that processes have enough memory located to them • Makes sure that two processes don't try to access the same memory location • Manage the transfer of pages between virtual memory and RAM • Allows multitasking
11(d)	• Interrupt

NOVEMBER 2023

uestion	Answer
3	**One** mark for each correct term:

Term	Description
hardware	A collective term for the physical components of the computer system.
application software	A type of software that provides services that the user requires and allows the user to perform tasks on the computer.
operating system	A type of software that manages the main functions of the computer, including managing files and managing memory.
firmware	A type of software that is stored in the read only memory (ROM). It includes the basic input output system (BIOS) and the bootloader.

(c)(i)	Any **two** suitable hardware example e.g.: – Moving the mouse – Clicking a mouse button – Plugging in a device – Paper jam in printer – Printer out of paper
(c)(ii)	Any **two** suitable software examples e.g.: – Division by zero – Two processes accessing the same memory location – Null value

NOVEMBER 2023 (3)

(d)	Any **three** from: – It **performs** the basic functions of a computer – It **manages** the hardware – It provides a platform to run software – It provides a user interface – It performs tasks such as (any example of function of an operating system)

MJ2024/11

6(a)	Any **one** from: • **Permanently** store instructions (in ROM) • Stores instructions to **boot up/start up** the computer • Provides the **operating system** with a platform to run on • Controls/manages/allows **communication** with hardware • **Store** instructions **securely** (to stop them being easily corrupted)
6(b)	Any **one** from: Example: • Bootstrap • Bootloader • BIOS • Operating system (in embedded system) • Programs (in embedded systems)
6(c)	Any **two** from: • Operating system • Utility software NOTE: Two examples of utility software can be awarded

MJ2024/13

9(a)	**One** mark for each correct term in the correct place. • printer • computer • priority level • fetch–decode–execute cycle • interrupt queue • higher • interrupt service routine (ISR)

9(b)	Example: • Memory management • Multitasking

FM2024/12

3(a)	**One** mark each:

Function name	Description
managing memory	Examples: • allocates memory to processes • prevents two processes accessing the same memory
platform for running applications	allows application software to run on the computer
managing peripherals	Examples: • allocates data to buffers • transmits data to hardware • receives data from hardware

3(b)(i)	To indicate that something requires the attention of the **processor/OS/CPU**

3(b)(ii)	**One** mark for input device **and** matching interrupt: e.g. • Keyboard Key pressed • Mouse Mouse moved//button clicked

3(b)(iii)	Any **five** from: • Interrupt is given priority • …and placed in interrupt queue • Processor finishes current **FE cycle** for program • Processor checks interrupt priority queue // processor checks for higher priority interrupt than program/process • If lower priority processor runs next FDE cycle for program/process // if lower priority processor continues with program/process • (if higher priority) processor **stores** current process/registers on stack • Checks source of interrupt • … and calls the appropriate ISR • ISR handles/resolves interrupt • If there is another higher priority interrupt (than process) then repeat • … (otherwise) processor retrieves content of stack/registers/previous process (to continue with process from program)

MJ2024/12

(c)	Any **two** from: • System software **manages/maintains** the **hardware/software** • Applications software allows the **user** to perform tasks

ON2024/12

1(a)	**Instructions/program** that is used to operate a computer/hardware
1(b)	B
1(c)	Operating system // system software

MJ2025/11

8(b)	Any **one** from: • Permanent **instructions/software** that are programmed into/stored in the ROM. • **Instructions/software** that allow hardware to be controlled/managed. • **Instructions/software** that provides the operating system with a platform to run on.
8(c)	Any **two** from: Examples: • user input // by example • encountering an obstacle • emergency stop • hardware error • software error • battery low • component overheating • two processes accessing the same location

9(a)	Any **three** from: Examples: • To make sure memory is used efficiently • Allocates/deallocates memory to processes • Checks that the processes being actioned have enough memory available • Moves data to and from memory/between memory and storage • Makes sure that two processes do not try and access the same memory location • Creates a memory partition • Creates virtual memory • Transmits pages between RAM and VM

4.2 Types of programming language, translators and integrated development environments (IDEs)

0478/11 – June 2019

7. – Syntax
   - High-level language
   - Translator
   - Machine code
   - Low-level language
   - Assembly

0478/11 – NOVEMBER 2019

Question	Answer	Marks
6(a)(i)	**One** from: ∞ Code will run without the need of an interpreter ∞ (Object) Code is platform independent ∞ Source code not available / cannot be modified	1
6(a)(ii)	**One** from: ∞ Source code not available / cannot be modified ∞ Comments, etc. not visible ∞ Future changes will require code to be recompiled	1

0478/13 – NOVEMBER 2019

Question	Answer	Marks
2(a)	**One** mark for each correct row	4

Statement	True (✓)	False (✓)
High-level languages need to be translated into machine code to run on a computer	✓	
High-level languages are written using mnemonic codes		✓
High-level languages are specific to the computer's hardware		✓
High-level languages are portable languages	✓	

2(b) **One** mark for the correct tick

Example program	Tick (✓)
`1011100000110000` `0000011011100010`	

`INP`
`STA ONE`
`INP`
`STA TWO`
`ADD ONE`	

`a = input()`
`b = input()`
`if a == b:`
`  print("Correct")`
`else:`
`  print("Incorrect")`
	✓

0478/12 – MARCH 2020

Question	Answer	Mark
4		5

Statement	Assembler (✓)	Compiler (✓)	Interpreter (✓)
Translates low-level language to machine code	✓		
Translates high-level language to machine code		(✓)	✓
Produces error messages	(✓)	✓	✓
Translates high-level language one line at a time			✓
Produces an executable file	(✓)	✓	

1 mark per each correct row:

NOTE: tick shown in brackets (✓) is optional

0478/12 – JUNE 2020

Question	Answer
2(a)	Any **four** from: – To translate the high-level language into low-level language – Interpreter used whilst writing the program – Interpreter used to debug code line by line – Compiler used when program completed – Compiler used to create separate executable file (so compiler no longer needed) – If it runs first time in a compiler there are no syntax errors
2(b)	Any **three** from: – Easier to understand // Don't know assembly code – Easier to debug – Easier to maintain – Portable – Knowledge of manipulating memory locations/registers not required – Can use an IDE – Greater range of languages

Question	Answer
7(a)	Any **two** from: – Makes use of words // close to human language – Machine independent // portable – Problem / logic focussed – Needs to be translated/interpreter/compiled (to low-level for processing by computer) // needs converting to machine code
7(b)	**Four** from **Max 2** for only giving compiler/interpreter features – **Both** translate high level / source code to machine code – **Both** generate error diagnostics/messages // identify errors – Interpreter translates one line at a time // checks one line and then runs it – Compiler translates whole code in one go // checks all code and then runs it – Interpreter **stops** when meets an error – …and then allows you to continue running from where you stopped // correct errors in real-time – Compiler provides list of all errors – Interpreter does not produce an **executable** file – Compiler produces an **executable** file

0478/13 – JUNE 2021

Question	Answer
5	One mark per each correct row

Statement	High-level language (✓)	Assembly language (✓)	Machine code (✓)
It requires a translator to be processed by a computer	✓	✓	
It is an example of low-level language		✓	✓
It uses mnemonics		✓	
It uses English-like statements	✓		
It can be used to directly manipulate hardware in the computer		✓	✓
It is portable	✓		

NOVEMBER 2021 (1)

8(a)	– High-level
8(b)(i)	**One** mark for the correct translator, **two** marks for the benefit(s). – Interpreter – Easier to debug – … as errors are immediately reported when detected
8(b)(ii)	**One** mark for the correct translator, **two** marks for the benefits. – Compiler – Creates an executable file – … so, translator is no longer needed to run it – Source code cannot be stolen // can be provided without the source code

NOVEMBER 2021 (2)

Question	Answer
6(a)	Any **one** from: – They both translate **high-level language** into **machine code / low-level language** – They both check for errors – They both report errors
6(c)	– Assembler

MARCH 2022

Question	Answer
7(a)(i)	1 mark for when e.g. • Development // when writing the program // when debugging 1 mark for explanation to max 2 from: e.g. • … easier to debug • …stops when an error is detected • …reports one error at a time • …can correct errors in run-time // correct the line and then continue running from that point • …can test one section without the rest of the code being completed
7(a)(ii)	1 mark for when e.g. • After completion // For distribution // For final/repeated testing 1 mark each to max 2 from: e.g. After completion • It creates an executable file • …than can be distributed without source code • …so that other people cannot edit/view the code • …so end users do not need translator software // so end users do not need to compile/interpret each time • …so it is machine/platform independent (usually) In final testing • It creates an executable file • …do not need to retranslate for each test sequence • …can test repeatedly with different data faster

MARCH 2023

Question	Answer
2(a)	No mark for choice. Any **four** from matching choice. High-level • Easier for programmer to read/write/understand/edit • … therefore, the programmer is less likely to make mistakes // can write in shorter timeframe • Easier to **debug** // Easier to find/correct **errors** • …so, the programmer can find and correct errors in less time • Game will be machine independent // Game will be portable (between hardware) • …the game can be used on any computer without a need for understanding of the hardware / compilation for that hardware • Programmer can focus on the problem instead of the manipulation of memory/hardware Low-level • More **memory/RAM** efficient • … 3D graphics will have high memory consumption anyway • Allows direct manipulation of memory • … allows for more efficient control/response time • Allows for use of specialised hardware

JUNE 2023

7(a)	Any **two** from: • Close to the language processed by computers • May use mnemonics • An example is assembly language/machine code
7(b)	Any **two** from: • Can directly manipulate the hardware • No requirement for the program to be portable • Program will be more **memory efficient** • No requirement for a compiler/interpreter • Quicker to execute • Can use specialised hardware

JUNE 2023 (2)

(c)	Any **three** from: e.g. • Code editors • Run-time environment • Built-in interpreter • Error diagnostics • Auto-completion • Auto-correction • Prettyprint

JUNE 2023 (3)

Question	Answer
4(a)	• B
4(b)(i)	• Machine code // low-level language // object code
4(b)(ii)	• Interpreter
4(b)(iii)	• Compiler
4(b)(iv)	• Compiler

MJ2020/11

9(a)	Any **three** from: – Closer to/is machine code – May use mnemonics – May need an assembler to be translated – One line of code represents a single instruction – Machine dependent – Have direct access to memory locations/registers
9(b)	– Assembly code – Machine code
9(c)	Any **one** from: – It is more difficult to understand – Error prone – Have to manipulate memory locations – Machine dependent

ON2022/11

7(a)	• Low-level language
7(b)	• Assembler
7(c)	Any **two** from: • He can directly access the hardware • He can use special machine-dependent instructions • There is no need for the program to be portable • Smaller file size // takes up less storage space • More efficient use of memory • Programs will be more time efficient when running
7(d)	Any **two** from: • Programs are not portable • It is complex to learn • Difficult to debug

MJ2023/12

5(a)	• C

(ii)	Any **two** from: • It creates an error report after **trying to compile** • … displaying **all** errors in the code • … that require correction before execution can take place

ON2023/11

9(a)	Any **one** from: – Operating system // Interrupt handler
9(b)	Any **five** from: e.g. – Key press generates the interrupt – Interrupt given a priority – Interrupt is sent to CPU – Interrupt is placed in a queue – CPU stops current task to check the queue/service the interrupt … – … using an interrupt service routine – If key press is highest priority the interrupt is processed

ON2024/11

(e)	**Six** from (MAX **4** for stating features): Code editor … … that allows the user to enter and amend code in their program Run-time environment … that allows a program to be run and see the outputs of their program Error-diagnostic … to show the programmer where there are errors in the program Auto-completion … to give the user **options/suggestions** of key commands to select Auto-correction … to correct a command that has a **minor misspelling** Prettyprint … changes the colour of **key commands** do they are easy to identify

ON2024/13

7	Any **four** from: • It is a piece of software • … used to write/develop/edit code • … used test/debug the code • … with features such as auto-completion // any suitable example • To translate the code to low level language/machine code

FM2025/11

7(a)	1 mark each: ● Language: assembly ● Translator: assembler
7(b)	Any four from ● Each statement is translated and executed before the next ● Translator stops when an error is found ● … can correct and continue from where stopped // real time changes ● ... does not need to retranslate all of the code again (unlike a compiler) ● Can test sections of the code … ● … without all of the program being functional/accurate/complete

MJ2025/12

3(a)	Any **two** from: • It is easier to debug. • Less likely to make errors. • The program is machine independent/portable
3(b)(i)	**One** mark for each correct term in the correct place. • whole code • executing • all • line by line • error 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.
3(b)(ii)	**One** mark for each correct function. **One** mark for each correct matching role description. Examples: • Code editor • Allows the programmer to write/change the program. • Run-time environment • Allows the user to run the code and see the output. • Error **diagnostics** • Features that can be used to find errors in the code • Auto-completion • A programmer starts to type a command word, and the IDE suggests **option** for completing it. • Auto-correction • If a programmer **misspells** a command word it is changed to the correct spelling • Prettyprint • The **command words/identifiers** are given different colours