7.3 Structure and role of the processor part 1

Q1.

All marks AO1 (understanding)

Level	Description	Mark Range
4	A line of reasoning has been followed to produce a coherent, relevant, substantiated and logically structured response. The response covers all three areas indicated in the guidance below and in at least two of these areas there is sufficient detail to show that the student has a good level of understanding. To reach the top of this mark range, a good level of understanding must be shown of all three areas.	10-12
3	A line of reasoning has been followed to produce a coherent, relevant, substantiated and logically structured response which shows a good level of understanding of at least two areas indicated in the guidance below.	7-9
2	A limited attempt has been made to follow a line of reasoning and the response has a mostly logical structure. At least four points have been made. Either a good level of understanding of one area from the guidance has been shown or a limited understanding of two areas.	4-6
1	A few relevant points have been made but there is no evidence that a line of reasoning has been followed. The points may only relate to one or two of the areas from the guidance or may be made in a superficial way with little substantiation.	1-3

Guidance – Indicative Response

For each guidance point, if the student expands on the point to explain in what way the measure will improve performance then this can be considered to be a second point. For example:

- "Using a processor with more cores" is one point.
- "Using a processor with more cores which will be able to execute multiple instructions simultaneously" is two points.

Note that just "faster" is not enough to count as an expansion point without an explanation of why.

1. Server Hardware

Replace the processor with one which has more cores

Replace the processor with one which has more cache memory // increase the

amount of cache memory

Replace the processor with one which runs at a faster clock speed **NE.** faster processor

Use a parallel processor architecture // use more processors which can work in parallel

Use a processor with a bigger word size

Use a processor that makes (better) use of pipelining

Install more RAM // main memory // primary memory

Use RAM // main memory // primary memory with a faster access time

Replace HDDs with SSDs // Replace HDDS with HDDs that can read data at a faster rate

Defragment the HDD

Replace the motherboard with one which has buses which run at a faster clock speed

Replace the motherboard with one which has more lines in the data bus

Use the Harvard architecture

Distribute the processing across multiple servers

2. Network

Replace the network cable with cable that has a higher bandwidth // replace copper cable with fibre-optic cable **A.** Ethernet cable for fibre-optic NE. higher bandwidth network

Replace any wireless / WiFi connections with wired ones

Replace the network cards with ones that can transmit data at a higher bitrate

Consider the overall network design eg how the network is divided into subnets **A.** split the network into subnets

Use a star topology (instead of a bus)

Consider using a more efficient protocol for the data across the network

Add additional wireless access points

3. Database and Software

Use a more efficient technique for controlling concurrent access to the database // replace record/table locks with serialisation/timestamp ordering/commitment ordering

Replace the database software with software that uses more efficient algorithms for tasks **A.** examples eg replace linear search with binary search

Use the index feature of the database to speed up searching on fields that are commonly used for this purpose

Rewrite the database software in a language that is suitable for concurrent execution // use a functional programming language for the database software

Ensure the software is compiled rather than executed by an interpreter // rewrite the software in assembly language/machine code

Review the conceptual model of the database to see if it contains any inefficiencies such as data redundancy that could be eliminated **A**. normalise the database design

Consider if it would be appropriate to sacrifice normalisation of the conceptual model to improve performance

Use a non-relational database system **A.** examples eg NoSQL

Distribute the data across multiple servers

Try to reduce the amount of other (unrelated) software that might be running on the database server at the same time

Try to reduce the number of database accesses that need to be made simultaneously // run some tasks at quiet times / overnight

Purge / archive data that is no longer necessary / in use

[12]

Q2.

(a) **Marks are for AO3 (program)**

Answer 1

1. ADD R0, R0, #1 ;
2. CMP R0, #11 ;
3. BNE; startloop ;

Answer 2

1. ADD R0, R0, #1 ;
2. CMP R0, #11 ;
3. BEQ endloop ;
4. B startloop ;

Answer 3

1. CMP R0, #10 ;
2. BEQ endloop ;
3. ADD R0, R0, #1 ;
4. B startloop ;

Answer 4

1. ADD R0, R0, #1 ;
2. CMP R0, #11 ;
3. BLT; startloop ;

Stop marking when the first incorrect command is encountered. Mark response against whichever alternative gives the highest mark.

I. Any extra commands which do not effect operation of program.

4

(b) **Mark is for AO2 (apply)**

28_{10} // $(000)11100_2$;

TO. If two answers given and one is incorrect.

I. Lack of subscript.

1

[5]

Q3.

Mark is for AO1 (understanding)

Direct addressing means that the operand is the (memory) address/register number (of the datum) whereas immediate addressing means the operand is the datum ;

Note: Must be clear that the operand is being used.

[1]

Q4.

Key points of subject criteria:

FETCH:
Contents of Program Counter/PC transferred to Memory Address Register/MAR;
Address bus used to transfer this address to main memory;
Contents of addressed memory location loaded into the Memory Buffer Register/MBR;
Transfer of content uses the data bus;
Increment contents of Program Counter/PC;
Increment Program Counter/PC and fetch simultaneously;
Transfer content of Memory Buffer Register/MBR to the Current Instruction Register/CIR;

A. Increment PC at any part of fetch process after transfer to MBR
A. Reference to MDR (memory data register) instead of MAR

DECODE:
Decode instruction held by the Current Instruction Register/CIR;
The control unit decodes the instruction;
Instruction split into opcode and operand(s);

EXECUTE:
If necessary, data is fetched;
The opcode identifies the operation to be carried out by the processor;
Execute instruction by relevant part of processor;
Result stored in accumulator/ (destination) register;
Status Register (SR) updated;
If jump/branch instruction Program Counter/PC is updated;

Mark Bands and Description

7-8	*To achieve a mark in this band, candidates must meet the subject criterion (SUB) and all 5 of the quality of written communication criteria (QWCx).* *SUB* Candidate has provided at least 7 points. At least one point made for each of the fetch, decode and execute stages. Answer must mention at least 3 registers. *QWC1* Text is legible. *QWC2* There are few, if any, errors of spelling, punctuation and grammar. Meaning is clear. *QWC3* The candidate has selected and used a form and style of writing appropriate to the purpose and has expressed ideas clearly and fluently. *QWC4* Sentences (and paragraphs) follow on from one another clearly and coherently. *QWC5* Appropriate specialist vocabulary has been used.
4-6	*To achieve a mark in this band, candidates must meet the subject*

	criterion (SUB) and 4 of the 5 quality of written communication criteria (QWCx). *SUB* Candidate has provided at least 4 points covering at least 2 of the fetch, decode, execute stages. Answer must mention at least 2 registers. *QWC1* Text is legible. *QWC2* There may be occasional errors of spelling, punctuation and grammar. Meaning is clear. *QWC3* The candidate has, in the main, used a form and style of writing appropriate to the purpose, with occasional lapses. The candidate has expressed ideas clearly and reasonably fluently. *QWC4* The candidate has used well-linked sentences (and paragraphs). *QWC5* Appropriate specialist vocabulary has been used.
1-3	*To achieve a mark in this band, candidates must meet the subject criterion (SUB) and 3 of the 5 quality of written communication criteria (QWCx).* *SUB* Candidate has made a small number of relevant points. *QWC1* Most of the text is legible. *QWC2* There may be some errors of spelling, punctuation and grammar but it should still be possible to understand most of the response. *QWC3* The candidate has used a form and style of writing which has many deficiencies. Ideas are not always clearly expressed. *QWC4* Sentences (and paragraphs) may not always be well-connected. *QWC5* Specialist vocabulary has been used inappropriately or not at all.
0	Candidate has made no relevant points.

[8]

Q5.

(a) Effective speed at which data can be retrieved will be increased;
A. Larger “chunks” of data/instruction can be fetched in one operation

Max 1

(b) The amount of available memory locations / addressable locations will double;
NE. increases amount of memory

1

[2]

Q6.

(a) **Marks are for AO1 (knowledge)**

instructions are stored in main memory;
instructions are fetched, (decoded) and executed by the processor;
programs can be moved in and out of main memory;
Max 2

MAX 2

(b) **2 marks for AO1 (knowledge) and 4 marks for AO1 (understanding)**

Level	Description	Mark Range
3	A detailed description, indicating a comprehensive knowledge has been provided which covers all three stages. For each stage, the description covers the majority of the points listed in the guidance. The answer is well structured and points are connected in a way that demonstrates a good understanding of the complete cycle.	5 – 6
2	An adequate description indicating knowledge of the cycle has been provided that either covers one or two stages in a good level of detail, including the majority of points for each stage, or covers all three stages but at a more superficial level. The answer is satisfactorily structured and points are connected in a way that demonstrates an understanding of some parts of the cycle.	3 – 4
1	A small number of points, from one or more stages have been recalled indicating some knowledge of the cycle. However, these have not been connected and demonstrates little or no understanding of any stage of the cycle.	1 – 2

FETCH:
- contents of PC transferred to MAR
- address bus used to transfer this address to main memory
- contents of addressed memory location moved into the MBR
- transfer of content used the data bus
- increment PC
- transfer content of MBR to CIR.

DECODE:
- decode instruction held by the CIR
- the control unit decodes the instruction
- instruction split into opcode and operand.

EXECUTE:
- if necessary, data is fetched
- the opcode identifies the instruction to execute / operation to perform
- execute instruction by relevant part of processor
- result stored in accumulator

6

(c) **Mark is for AO1 (knowledge)**

1 mark: A language that is very similar to / based upon the instruction set of the computer;

1

(d) **Mark is for AO1 (knowledge)**

1 mark: (opcode) represents the instruction to be executed;

1

(e) **1 mark for AO1 (knowledge) and 1 mark for AO1 (understanding)**

AO1 (knowledge):

1 mark: Immediate addressing:
the operand value is part of the instruction / / no need to go to any memory address;

AO1 (understanding):

1 mark: Example:

MOV RX, #Y;

[where X is 0-12 and Y is a decimal value]

2

(f) **1 mark for AO3 (design) and 3 marks for AO3 (programming)**

```
    CMP R1, #5          compare r1 against 5
    BNE endif           jump to end of statement
                        if not equal
    MOV R2, #10         move the value 10 to B
endif:
```

AO3 (design) – 1 mark:

1 mark: Identifying that a comparison and branch are required to have the same effect as the IF statement, even if the syntax or comparison made are incorrect

AO3 (programming) – 3 marks:

For the AO3 (programming) marks, the syntax used must be correct for the language as described on the question paper.

1 mark: Comparing R1 against 5 and having a branch with the correct logical condition
1 mark: For moving 10 to R2
1 mark: For having a label for end of statement (that is used in the branch)

I Load instruction to setup R1 from X.

I Store instruction to store R2 into B.
A labels given in any sensible format
DPT - missing hash for immediate addressing

4

[16]

Q7.

(a) operand;
R operand code

1

(b) (i) PC 0010;
MAR 0001;
MBR 00100100;

3

(ii) The instruction is held in the CIR // instruction in CIR is decoded;
A IR

The control unit / instruction decoder decodes the instruction;
NE the processor decodes the instruction

Instruction will be split into opcode and operand;
R if it is implied that a register will do this splitting / decoding

Relevant part of processor / CPU executes instruction // using ALU to perform calculations;
A instruction executed by the control unit / ALU
NE processor executes instruction

Further memory fetches / saves carried out if required;

Result of computation stored in accumulator / register / written to main memory;

Status register updated;
If jump / branch instruction PC is updated;

By example:
Will ADD contents memory location 0100 to accumulator;

MAX 3

(c) The current value in the accumulator would be stored in (memory) address / location 0011 / 3;

Number 011 / 3 stored in (memory) address / location 0011 / 3;

MAX 1

[8]

Q8.

(a) increase the number of bits that can be transferred <u>at one time</u> ;
A increase rate of data transfer;

increases the number of (memory) addresses / addressable locations / / increase the maximum amount of primary store / memory (possible);

instructions performed more quickly / / instructions executed at faster rate / /
fetch execute cycle will happen faster / /
increased heat may cause malfunctioning of device / / overheating;

A calculations / operations / commands for instructions

3

(b) (i) a (hardware) device / component that is not part of the CPU;
NE processor / computer
a (hardware) device not directly under the control of the processor / CPU;
a device that communicates through an I / O controller;
external hardware / device;
R examples alone

MAX 1

(ii) to allow exchange of data / instructions / signals between the processor and the peripheral;
A communicate
R information
NE To allow the device to be connected

1

(iii) Electronics that interface the controller to the system bus;
Electronics appropriate for sending signals to the device connected to the computer;

MAX 1

(iv) Each peripheral operates in a different way;
Not sensible to design a processor to control every possible peripheral;
A new type of peripheral would require the processor to be redesigned;
Peripherals may operate at a different voltage from the processor;
Peripherals will usually operate at a slower rate than the processor (requiring buffering);

MAX 2

[8]

Q9.

(a) Program Counter / Sequence Control Register;
Memory Address Register;
Memory Buffer Register / Memory Data Register;
Current Instruction Register;
R Abbreviations

Max 2

(b) **Step 1:** MAR ⟵ [PC] / Contents of program counter transferred to MAR;

R MAR ⟵ PC

R [MAR] ⟵ PC (see note about **DPT**)
R PC sends / transfers

Step 2b: MBR ⟵ $[Memory]_{addressed}$ / Contents of addressed memory location loaded into MBR; (must have concept of data coming from address in memory, not just going into MBR)

Step 4: Decode instruction;

A Contents of CIR decoded
A Instruction is split into opcode and operand
R Data for instruction
R CIR decoded, CIR decodes instruction
Note: A. [CIR] decoded

1 mark for each correct step

For PC accept Program Counter / SCR / Sequence Control Register
For MAR accept Memory Address Register
For MBR accept Memory Buffer Register / MDR / Memory Data Register

A Other means of indicating transfer e.g. [PC] ⟶ MAR
A [Memory] for $[Memory]_{addressed}$
DPT – no / incorrect square bracket use for register transfer notation

3

[5]

Q10.

(a) A set of rules / regulations (to allow communication between devices) // set of agreed signals / codes for data exchange;
NE a rule // a regulation // a signal // a code
NE instruction(s)

1

(b) Analyses statement by statement each line of source code
A runs / translates / executes line by line
R compiles (line by line)

Calls routines to carry out each instruction / statement

Max 2

(c) Instructions / programs stored (with data) in main memory; **A** memory // RAM

Program run by fetching, (decoding and executing) instructions (from main memory)* in sequence;

Program can be replaced by loading another program into (main) memory

Contents of a (main) memory location can be interpreted as either an instruction or data;

* = This mark can be awarded without the explicit reference to main memory if main memory has already been mentioned elsewhere in the response.

Otherwise, the answer must make clear that the instructions are coming from the main memory to get this mark.

3

(d) LOAD 21
STORE 23

LOAD 22
STORE 21

LOAD 23

STORE 22

1 mark for value from 21 stored into 23;
1 mark for value from 22 being moved to 21;
1 mark for value from 23 being moved to 22;

Alternative :

LOAD 22
STORE 23

LOAD 21
STORE 22

LOAD 23
STORE 21

1 mark for value from 22 stored into 23;
1 mark for value from 21 being moved to 22;
1 mark for value from 23 being moved to 21;
DPT if a different temporary storage area is used
I end of statement separators
Max 2 if the program does not fully work

3

(e) Robots find it hard to adapt to changes in environment // Robots are unable to adapt to changes easily;

Robots find it hard to work with 3D vision;

Robots find it hard to detect edges between similar objects // robots find it hard to perform shape detection;

Robots find it hard to get feedback when gripping items;

Robots find it hard to pick up balls // ball difficult shape to grip // balls can roll away;

Robots have limited processing power // too many variables to deal with;

Programming for vision/grip is a complex problem;
A child builds up experience of using touch / vision;

A Robot cannot recognise when it makes mistakes;
A Robot can't think for themselves // can't perform lateral thinking

Max 3

(f) (i) (Lens focuses) light / photons onto image sensor;
R if uses 'reflection'

Image sensor is a CMOS / CCD / photoelectric device;
CCD used ADC to convert measurement of light intensity into binary;
CMOS uses transistors to generate binary value;
Image sensor converts light into discrete / electrical signals / binary;

Image is captured when the shutter is pressed;
Large pixels collect more electrons than small pixels and so produce better quality images;

Firmware performs data processing to “tidy up” image;
(Colour) filter used to generate data separately for Red, Green, Blue colour components;
Aperture / shutter speed can be adjusted to cope with varying lighting conditions;
Image is recorded as group / array of pixels // Image sensor consists of array of pixel (sensors)//etched into the image sensor’s silicon are pixels;

Image data transferred to robot;
Image data usually stored on solid-state disk;

Max 3

(ii) Robot has a low powered microprocessor;

Too much image data for the robot to process quickly // smaller resolution can be processed quicker;

A high resolution image has too much image data for the robot to store // low resolution uses less storage space;

Do not need high resolution to determine colour of balls;

NE allows more images to be stored

Max 1

[16]

Q11.

(a) A set of / group of / parallel wires / lines;

Wires needs to be qualified with set / group

that are used to connect together components (inside the computer) // connect different parts of the CPU

in order to pass signals between them;

R a wire
A connect different parts of the computer
NE data

Max 2

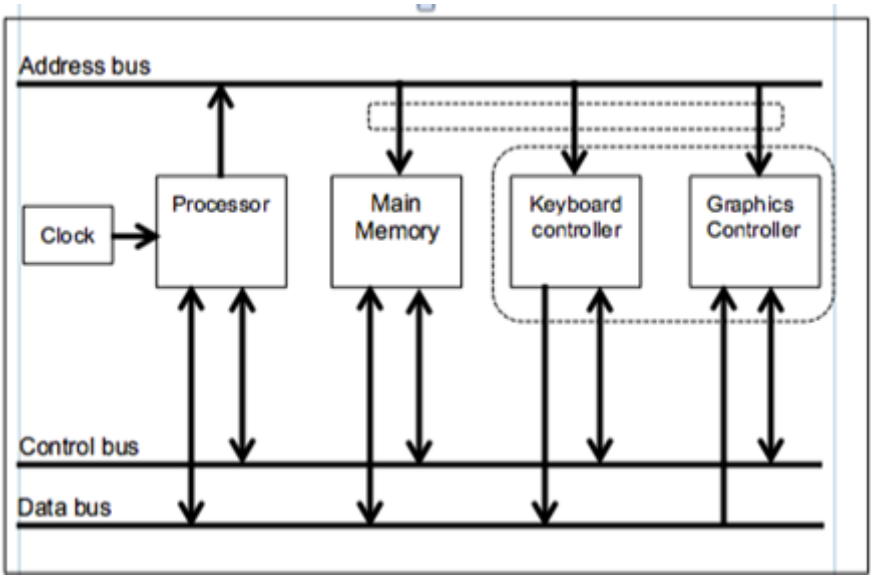
(b) Instructions;
A Commands / machine-code
R signals

Examples of a control signal (Max 1):
NE an event that details when an interrupt would be caused

Clock / timing; reset; interrupt ACK; interrupt request; bus grant; bus request; status; I / O write; I / O read; memory read; memory write; transfer ACK
A interrupt **A** transfer request
A read / write
NE load / store
NE clock speed

2

(c)



1 mark – one of processor, keyboard controller
or graphics controller identified correctly
2 marks – all three correctly identified

Address bus connects the 4 components;
Arrow from processor to the address bus;
Arrows from address bus to the three other components;

Mark this on where the candidate has put the components.

5

[9]

Q12.

(a) (i) Indicates the basic machine operation / function / command;
Executable binary code;
A "instruction" – with a valid example

Max 1

(ii) Represents a single item of (binary) data / a single value;
Represents a memory address / storage location;
The value that the instruction operates on;
A parameter for the operation
NE "address"

Max 1

(b) Easier to understand;
Takes less time to code (as using mnemonic opcodes and hex operands);
Fewer mistakes made in coding;
Ability to add comments to code;
Use of symbolic names for operands // easier to remember opcodes / mnemonics;
Use of labels;
Easier to maintain / debug;
NE easier to read / code / write
NE quicker
A converse points if clearly discussing machine code

Max 2

[4]

Q13.

Key points of subject criteria:

Fetch:
Contents of Program Counter / PC transferred to Memory Address Register / MAR;
Address bus used to transfer this address to main memory;
Contents of addressed memory location loaded into the Memory Buffer Register / MBR;
Transfer of content uses the data bus;
Increment contents of Program Counter / PC;
Increment Program Counter / PC and fetch simultaneously; **A** any part of fetch process
Transfer content of Memory Buffer
Register / MBR to the Current Instruction
Register / CIR

Decode:
Decode instruction held by the Current
Instruction Register / CIR;
The control unit decodes the instruction;
Instruction split into opcode and operand;

Execute:
If necessary, data is fetched;
The opcode identifies the type of instruction it is;
Execute instruction by relevant part of processor;
Result stored in accumulator;
Status register updated;
If jump / branch instruction Program Counter/PC is updated;

To achieve a mark in this band, a candidate must meet the subject criterion (SUB) and 4 of the 5 quality of language criteria (QLx).

SUB Candidate has provided at least 6 points. At least one point made for each of the fetch, decode and execute stages. Answer must mention at least 3 registers.
QL1 Text is legible.
QL2 There are few, if any, errors of spelling, punctuation and grammar. Meaning is clear.
QL3 The candidate has selected and used a form and style of writing appropriate to the purpose and has expressed ideas clearly and fluently.
QL4 Sentences and paragraphs follow on from one another clearly and coherently.
QL5 Appropriate specialist vocabulary has been used.

6

To achieve a mark in this band, candidates must meet the subject criterion (SUB) and 4 of the 5 quality of language criteria (QLx).

SUB Candidate has provided at least 4 points covering at least 2 of the fetch, decode, execute stages. Answer must mention at least 2 registers
QL1 Text is legible.
QL2 There may be occasional errors of spelling, punctuation and grammar. Meaning is clear.
QL3 The candidate has, in the main, used a form and style of writing appropriate to the purpose, with occasional lapses. The candidate has expressed ideas clearly and reasonably fluently.

QL4 The candidate has used well-linked sentences and paragraphs.
QL5 Appropriate specialist vocabulary has been used.

4–5

To achieve a mark in this band, candidates must meet the subject criterion (SUB). The quality of language should be typified by the QLx statements.

SUB Candidate has provided at least one valid point.
QL1 Most of the text is legible.
QL2 There may be some errors of spelling, punctuation and grammar but it should still be possible to understand most of the response.
QL3 The candidate has used a form and style of writing which has many deficiencies. Ideas are not always clearly expressed.
QL4 Sentences and paragraphs may not always be well-connected or bullet points may have been used.
QL5 Specialist vocabulary has been used inappropriately or not at all.

1–3

Candidate has not made reference to any of the points above.

0

[6]

Q14.

(a) Third (generation) // 3;
R High Level Language

Do not reject high level language if answer also contains '3rd generation' – refer upwards for anything else.

1

(b) (i) Hexadecimal // base 16;
A Hex

Hex used in textbook

1

(ii) Take up less space when printing / viewing;
NE takes up less space
Less likely to make errors;
Op-codes are easier to recognize;
Easier to understand;
Less time taken when coding as more concise // quicker to program;
NE easier to read
NE quick to write

Max 1

(iii) Lowest address : 00
Highest address : FF

BOTH correct to gain one mark;

A 0 for lowest address
A 255 for highest address
A notation in front of hex &, $

1

(c) When coding for execution speed;
When coding to minimize object code size;

When writing code to control devices / directly access hardware;
A When coding for a specific processor;
A by example if maps to one of the above

Max 1

(d) A compiler produces object code/machine code;
whilst an interpreter does not produce any object code;
Interpreted code will execute slower;
than executing the object code produced by a compiler;
You always need the interpreter to interpret source code;
but you do not need the compiler to execute a compiled program;
Once compiled source code is no longer required to run the program;
An interpreter always needs source code at runtime;
Compiled code can only be executed on a machine with the same processor type / instruction set;
Interpreted code is more portable;
A compiler translates the whole source code (at once);
An interpreter analyses the code line by line;
NE reads

Max 4

[9]

Q15.

(a)

Number	Component
1	Memory address register; **NE MAR;**
2	Data bus;
3	Control bus;

3

(b) To fetch / decode / execute instructions;
To synchronise operation of processor;
To marshal/control operation of fetch-execute cycle;
To send control signals/commands to other components of fetch-execute cycle;
To control the transfer of data between registers/MBR;
A by example
NE information

Max 1

(c) Arithmetic (and) logic unit;
NE Arithmetic unit
NE Logic unit

1

(d) A (very fast) memory location within the processor;
A A (very fast) memory location within an I/O controller;

1

(e) Arithmetic results – Overflow/underflow/positive/negative/zero/carry;

Interrupts (enabled/disabled);
Parity;
BCD arithmetic enabled/disabled;
Supervisor mode;
Halt;
A illegal instruction/operation
Refer to team leader with other potentially correct answers.

Max 1

[7]

Q16.

(a) Second (generation);
A 2
R assembly code / language

Note: Adding "assembly" / "assembler" does not talk out a valid mark for second / 2

1

(b) (memory) Address / location / offset;
A line number
R instruction number

1

(c) (y) Opcode / operation code;
A op-code
NE operation
(z) Operand;

2

(d) **Individual Instructions:**
One to one / each assembly language instruction translates to one machine code instruction;

Programs:
Figure 1 assembly language equivalent of figure 2 // figure 2 machine code version of figure 1 // figure 2 is assembled version of figure 1;
NE figure 2 "binary version" of figure 1
NE different generations of language

1

[5]

Q17.

(a) 1 – clock;
2 – (Main) memory / IAS; **A** RAM **R** ROM
3 – Control bus;
4 – VDU controller / output controller; **A** controller for other named output device
5 – Processor; **R** Central Processing Unit / CPU

5

(b) Memory address register;
R abbreviations

1

(c) Memory buffer register / memory data register;

R abbreviations

1

(d) Address bus has 64 lines / tracks/ wires // there are 2 ^64 memory locations available;
NE 64 bits wide, moves 64 bits of data

1

[8]

Q18.

(a) Address (bus);

1

(b) 1;
R 33

1

(c) A – Visual display unit; **A** VDU
B – Processor; **R** CPU
C – (Main) memory;
D – Keyboard;

4

[6]

Q19.

(a) Operand – 5
Opcode – LOAD ;
A binary value 101 with any number of preceding zeroes for the operand

Both needed for the mark

1

(b)

```
LOAD   7;
ADD    8;
ADD    3
STORE
          21
          ;
```

Both Add instructions for the mark - do not need to follow each other.

The operands for LOAD and ADD can be in any order
I an end of line indicator symbol e.g. ";"
I comments explaining code
I additional unnecessary commands
R commands with a # or ( ) or [ ] in the operand
A operands in binary
A operands in binary and opcodes in binary, if candidate has provided a translation table
A correct operands in hex if using &

Max 2 if code would not produce correct result

3

[4]

Q20.

(a) Number	Component Name
1	Memory Address Register
2	Address Bus
3	Memory Data / Buffer Register
4	Data Bus

4

(b) The instruction is held in the CIR;
A IR
The control unit / instruction decoder decodes the instruction;
The opcode identifies the type of instruction it is;
Relevant part of CPU / processor executes instruction;
A ALU
Further memory fetches / saves carried out if required;
Result of computation stored in accumulator / register / written to main memory;
Status register updated;
If jump / branch instruction, PC is updated;
A SCR

Max 3

(c) Can be displayed in less space;
R takes up less space **NE**
Easier to remember / learn / read / understand;
Less error prone;

Max 1

(d) (i) Assembler;

1

(ii) HLLs are problem oriented;
HLL programs are portable // machine / platform independent ;
English like **keywords / commands/ syntax / code**;
R closer to English
Less code required // less tedious to program //
one to many mapping of HLL statements to machine code commands;
Quicker/easier to understand / write / debug /learn / maintain code;
R just quicker/easier
HLLs offer extra features e.g. data types / structures // structured statements // local variables // parameters // named variables/constants;
R procedures / modular
A example of a data structure
NE "extra features" without example
Speed of execution not crucial for most tasks so faster execution of assembly language not required;
Most computer systems have a lot of (main) memory / RAM so compact object code not essential;
A converse points for Assembly Language

3

[12]

Q21.

(a) Program Counter;
A Sequence Control Register
R Next Instruction Register
Current Instruction Register;
A Instruction Register
Memory Buffer Register;
A Memory Data Register
Memory Address Register;

Max 2

(b) Address in MAR/address to fetch instruction from, sent down Address Bus to Main Memory;
R address in PC (program counter)
Contents of address accessed in Main Memory;
A by implication if
contents of address location referred to during data transfer
Contents of address location//instruction//data passed down Data Bus into MBR/to processor;
A MDR instead of MBR
A RAM for Main Memory

Max 2

(c) Order of execution unimportant/one step does not rely on prior completion of the other;
Steps carried out by different (hardware) devices/components;
A operations are independent
A operations use different registers
R using different buses

Max 1

[5]

Q22.

(a) (i) LOAD = Opcode
4 = Operand

1 mark for both parts correct

1

(ii) A storage/memory location in the processor;
A CPU
NE location in the processor

1

(b) LOAD 12;
ADD 13;
STORE 14;
A operands 12 and 13 swapped around BUT NOT swapped opcodes
A correct binary operands 12- 1100 13- 1101 14- 1110
A minor spelling errors in Opcode only
P1 for use of # or other symbols with operand

Penalise each additional unnecessary instruction (beyond 3) by 1 mark

3

[5]

Q23.

(a) Smallest ;
picture element // unit which can be drawn on screen //
addressable / resolvable part / unit of a picture ;

2

(b) (i) 0010 1010 ;

1

(ii) 184 ;

1

(c) (i) pixels are stored as numbers // bit patterns / binary code // RGB bits ;

1

(ii) 8 ; **A** 1 byte

1

(d) (i) drawing is made up of drawing objects // or by example e.g. drawing is made up of circle / rectangle / straight line / etc. (must give at least two example objects) ;
different objects(**A** shapes) have a defined set of properties // or by example;

objects are stored as drawing commands / drawing list ;
some properties use mathematical equations / formulae ;

Max 2

(ii) object type ; co-ordinates / location of the centre **R** centre (only) ; radius / diameter ; fill colour ; fill style ; line thickness ; line colour ; line style ; anything reasonable ;
R colour (only) Position (only)

Max 3

[11]

Q24.

(a) **Step 1**: MAR ←[PC] / Contents of program counter transferred to MAR;

Step 2b: MBR ← $[Memory]_{addressed}$ / Contents of addressed memory location loaded into MBR; (must have concept of data coming from address in memory, not just going into MBR)

Step 4: Decode instruction;
A Contents of CIR decoded
R Data for instruction
R CIR decoded, CIR decodes instruction

1 mark for each correct step

For PC accept Program Counter / SCR / Sequence Control Register
For MAR accept Memory Address Register
For MBR accept Memory Buffer Register / MDR / Memory Data Register

For CIR accept Current Instruction Register / IR / Instruction Register
A Other means of indicating correct transfer e.g. [PC] → MAR or MAR:=PC
A Missing square brackets or alternative types of brackets
A Answers that miss out reference to "contents of"
A [Memory] for $[Memory]_{addressed}$

3

(b) (i) Increases the number of bits (**A** amount of data) that can be transferred at one time // increase rate of data transfer;

1

(ii) Increases the number of memory addresses / /Increase the maximum amount of primary store/memory (possible);

1

(iii) Instructions performed more quickly // Instructions executed at faster rate;
A Calculations for instructions (this time only)
A Operations for instructions
NE Speeds the computer up
R Processes, tasks for instructions

1

[6]

Q25.

(a) Load B;
Add #5;
A absolute addresses instead of A and B
Store A;

3

(b) (i) Assembler;

1

(ii) Compiler;
R Interpreter

1

[5]

Q26.

1. address of next instruction to be executed/fetched;
2. (contents of Program Counter) copied into Memory Address Register;
3. Contents of Program Counter incremented (by 1);
   A incrementing by more than 1
4. ...at the same time....; *(only give a mark if between correct statements)*
5. instruction/data held at that address is placed in the Memory Buffer Register;
6. Contents of Memory Buffer Register copied into Current Instruction Register;
7. Instruction held in Current Instruction Register is decoded;
8. If necessary data is fetched;
9. (and) instruction is executed by processor/ALU;
10. Address sent/transferred over address bus;
11. Data/instruction transferred to processor on data bus;
12. Result stored in accumulator;

Max 6

[6]

Q27.

(a) (memory) address / location;
R Line number

1

(b) second (generation) //assembly language/code/program //
2 / 2nd;

1

(c) (i) assembler;
R. assembly

1

(ii) error list / error report / error count / **A** error message / highlight statement(s) illegally formed / instruction count // symbol table;
R error

1

(d) program (instructions are) transferred from backing store to main memory;
program consists of a sequence of instructions;
stored in a (continuous area of) main memory;
an instruction is fetched (and decoded);
and then instruction executed (by the processor);
program can be replaced by another program at any time;
program instructions are treated as data;

Max 4

[8]

Q28.

(a) Need to access/address registers/exact memory addresses/ hardware directly;
Fast speed of operation needed;
Code needs to take up little memory// minimise the size of the executable code;

A no compiler/interpreter exists yet for the machine// no other translator exists;
R manipulate bits
R comparison with machine code

Max 2

(b) Takes longer to program;
Leads to more errors // more difficult to detect errors;
Requires more skill;
Difficult to understand;
Difficult to maintain;
Processor dependant// not portable// not problem oriented;

Max 1

[3]

Q29.

(a) **Processor/CPU;**
Explanation faster execution of (program) instructions / the fetch-execute cycle is faster;
R more 'calculations per second'
Simultaneous processes possible / duel/quad – core processor;

Max 2

Additional processor;
Processing is shared between two processors;

Graphics Card;
Explanation – increasing the speed at which images are rendered;

(main) memory / RAM;
Explanation – reduces main memory to disc transfers;
Fit memory which has a faster read/write speed;

R Clock
A Explanation – increasing the clock speed/over-clocking;
R Cache
A Explanation – program instructions are fetched faster from cache than main memory;

(b) **Secondary storage/memory/disc store** // **(external) hard disk**;
A HDD/ Hard drive
Explanation – the storage space/capacity is increased;
R: 'bigger hard drive' or similar

2

(c) **Hub device / USB ports**;
Card with additional serial /parallel ports / PCMCIA / USB ports;
R Card with additional I/O ports
Explanation – will allow/support the simultaneous connection of several devices;

Max 2

[6]

Q30.

(a) (i) 1 GB ;

1

(ii) 300 GB ;

1

(b) Control (bus) ;

1

(c) Data bus has to transport data values to and from various devices /internal components ;
Only the processor assigns address values to the different devices ;

Max 2

(d) Logical // read // write // jump/branch // input // output // data transfer ;
A Boolean

1

(e) Program instructions are transferred from backing store to main memory ;
Program consists of a sequence of instructions ;
Program is stored in main memory ;
and can be replaced by another program at any time ;
Instructions are fetched (in sequence) ;
Decoded ;
and then executed ;

Max 3

[9]

Q31.

(a) To perform mathematical operations/calculations;

1

(b) Fetches, decodes and executes instructions; To control/co-ordinate the other parts of the processor;

Max 1

(c) Accumulator;
Index Register;
Base Register;
Stack Pointer;
Current Instruction Register; **A** Instruction Register
Program Counter/ Instruction Pointer/ Sequence Control Register;
Memory Address Register;
Memory Data Register// Memory Buffer Register;
Flags Register// Status Register// Condition Code Register;
Interrupt Register;

Max 3

[5]

Q32.

Component	Name
1	Program Counter
2	Memory Address Register; **A** MAR
3	Address Bus;
4	Data Bus;
5	Memory Data Register/ Memory Buffer Register; **A** MDR/MBR
6	Current Instruction Register; A Instruction Register/IR/CIR

[5]

Q33.

(a) Halve the time to perform an operation; **A** Operations performed more quickly;

1

(b) Increase the number of bits transferred at any one time from 16 to 32// Double the number of bits transferred at any one time;

1

(c) Increase the number of memory addresses; from 2^{24} to 2^{32};

2

[4]

Q34.

(a) Assembly language/code/program // second (generation);

1

(b) Machine code // first (generation);

1

(c) (Memory) address / location;
R Line number

1

(d) Assembler;

1

(e) 1-to-l (mapping between instructions/op code/numbers written in assembler and their machine code equivalent) / each assembly instruction translates into one machine code instruction

1

(f) Error / error list / error report / error count / highlight statement(s) illegally formed // / instruction count // symbol table;

1

[6]

Q35.

(a) Program Counter; Sequence Control Register/Instruction Pointer
Instruction Register// Current Instruction Register;
Memory Buffer Register// Memory Data Register;
Memory Address Register;
Penalise initials once only

4

(b) (i) Test for an interrupt/ check priority of interrupt
Identify the source of the interrupt;
Save and/or restore the volatile environment/registers;
Service the interrupt; A handle the interrupt
Disable (lower priority) interrupts

Max 2

2

(ii) Placed between Execute and Fetch;
A before Fetch/ after execute R at end of cycle

1

(c) (i) Interrupting device/ source supplies;
An offset/vector;
A index/indexed address added to the base address;
A base register

Gives the start address of interrupt service routine/ ISR//
Address vector table cell contains start address of ISR/
R Interrupting device supplies start address of ISR

3

(ii) A different routine can be easily introduced//routine can be relocated/ dynamically loaded; *or words to this effect*
A The interrupting device only needs to supply a new offset

1

[11]

Q1.

A very good range of responses was received to this question, with approximately half of students achieving five or more marks. Most students addressed all three aspects of the question (hardware, network, database and software). Students tended to make more points about how the hardware could be improved than about the other two areas. This was acceptable but students needed to have covered all three areas to achieve a mark of ten or above.

Some students wrote too vaguely to achieve marks, for example by writing that a "faster processor" would improve performance, without referencing a factor such as the clock speed that would make the processor faster. Other mistakes included believing that the question required students to contrast thin-client and thick-client and that the system was web based.

A small number of students wrote about issues which might be causing the system to perform poorly instead of explaining how the performance of the system could be improved. Such responses were not worthy of a mark.

Q2.

(a) Answers to the assembly code question were generally better than the previous series, although more than half of students were not able to secure a mark. It was pleasing to see that previous advice has been heeded as far fewer students were using terminology that can be identified from the Little Man Computer, with the majority having instructions which were obviously AQA assembly code. Most commonly low-end marks were lost due to incorrect syntax – missing a comma In the `ADD` instruction is enough to lose the mark as syntactic correctness is essential. Whilst most students that scored well identified that a comparison was needed, getting the correct value for the comparison (`#10` or `#11`) combined with the correct termination criterion proved harder and students are encouraged to work through a number of practical tasks using a simulator if possible.

(b) A good number of students were able to provide a correct answer to this question. Although the answer was intended to be in denary, binary answers were not precluded in this question so either received the mark. Students should not rely on this in future series and should be able to answer in either base.

The most common mistake was 700, indicating the student was aware of what a shift was, but not the correct way to implement this.

Q3.

Whilst many students showed some understanding on this question, it was quite common for the mark to be dropped when it wasn't clear that they were referring to the operand. "Immediate addressing uses a value whereas direct addressing uses a memory address" is insufficient while "Immediate addressing uses the operand as a value whereas direct addressing uses the operand as a memory address" would suffice.

It seems quite common for students to believe an immediately addressed operand can only be an integer value. Whilst this could possibly be inferred from the instruction set used by AQA in programming questions, it would be beneficial for students' wider knowledge for them to understand the operand could refer to any binary data of any form.

Q7.

This question was based around the processing of instructions. The majority of students correctly identified operand as the answers to part (a). Part (b)(i) was looking at the fetch part of the fetch execute cycle and a lot of students secured full marks for identifying the values that would be in the registers. A group of students put the answer 0002 into the PC and had not spotted that the question asked for the answer to be written in binary.

Part (b)(ii) asked students to discuss the decode and execute stages. Good answers pointed out the need for the control unit to decode the instruction. Answers that implied that the CIR did the decoding itself were not credited as the CIR is only a register. Students who then identified that a certain part of the processor did the executing secured a mark and a mark was also awarded to those that discussed the role of the ALU.

There was confusion from some students as to the role of the ALU and the accumulator. Students who identified that the status register would be updated picked up a mark.

Q8.

(a) The content of this part has appeared in a similar format in a previous paper and a third of students achieved 2 or 3 marks for this question part. Candidates who secured marks could clearly distinguish what performance changes would occur in a way that was precise and tied in to how a computer works. Answers such as 'the computer would be faster' did not gain a mark when thinking about increasing the clock speed as this does not contain the depth that is expected from an AS-level candidate. Discussing how more instructions would be executed per unit time did secure the mark. Candidates also seemed confused over the address bus with answers along the lines of being able to get more memory addresses per unit time rather than discussing that an increase in the width of the address bus would lead to the CPU being able to access more addressable memory locations.

(b) This part was based around I / O controllers, which is an area of the specification that has not been asked about previously. It was obvious that candidates did struggle with this part and this could be due to it being a topic that has not actually been taught or a topic that has only been covered very briefly.

(i) A lot of candidates could describe what a peripheral was and secured the mark for this section. Candidates who talked about a component outside of the processor did not secure the mark as this could also include main memory which is not considered to be a peripheral.

(ii) A group of candidates could discuss the role of the I / O port in terms of allowing data to be transferred from the peripheral to the CPU. Others talked about allowing communication between peripheral and CPU and this also secured the mark.

(iii) As this topic proved to be hard not many candidates could describe another part of the I / O controller. Those that could generally talked about electronics allowing the connection to the system bus or considered the electronics necessary to connect the peripheral to an actual physical port.

(iv) In this part was a question that candidates attempted and many successfully secured marks. A popular reason, that was allowed for a mark, was the idea that it might slow down the processor. It was pleasing to see stronger candidates talk about the idea of a peripheral operating at a slower speed than the processor and some talked about the need for a buffer. The idea of a peripheral working at a different voltage to the processor was another popular reason that secured a mark.

Q9.

Last year the long answer question was based around the fetch execute cycle and this question covered a similar topic.

Nearly 80% of students gained all of the marks for part (a) and could correctly provide two registers that would be used in the fetch part of the cycle. Students who failed to secure marks often gave answers such as 'memory bus register' or just 'memory register' and perhaps did not recognise that they could have used the acronyms provided in the figure of part (b).

Part (b) proved to be harder than expected and the common problem was that answers failed to be precise enough to gain marks. When using register notation, marks were not awarded for answers such as MAR⟵ PC. The PC part needed to be written as [PC] as this indicates 'the contents of the program counter'. Students who provided written answers commonly indicated that registers actually 'moved' or 'fetched'. Answers such as 'the memory address register fetches…..' or 'the program counter passes…' are not accepted as credit worthy and students should remember that registers are only temporary holders of data. Another common mistake was the idea that the program counter holds instructions rather than the correct concept of it holding a memory address.

For the second mark the question was looking for the idea of an area in main memory being addressed and the contents of this location being passed back to the MBR. A common answer was just MBR ⟵ [MAR] which does not include actually going out to main memory. The majority of students secured at least one mark for question part (b) but it was not common for a student to secure full marks.

Q10.

This question asked students about a variety of topics all linked back to the idea of robotics. Over half of all students correctly provided a definition for protocol and the clearer answers linked this to the idea of an agreed set of rules to allow communication between devices. Some students who failed to secure the mark answered along the lines of instructions and programs rather than the idea of communication.

Part (b) asked students to identify how a HLL interpreter works. It was perhaps surprising that only half of the students managed to secure at least one mark on this question. It is clear that students got confused with the differences between a compiler and an interpreter with, some students answering this question by stating that it would 'compile'. Answers that just stated that 'it would interpret code….' also failed to secure marks. How an interpreter works beyond just translating code line by line is clearly not well understood and perhaps is an area centres could be encouraged to look at further.

Part (c) asked about the stored program concept. As a topic included in the name of the examination unit it was surprising to see that less than half of the students secured a mark on this question part. Of the credit worthy points made, it was common to see the idea that instructions are stored in the main memory of a device. A few students then went on to correctly identify that instructions are then fetched and executed by the processor. It was pleasing to see some students then discuss that the stored program concept allows different programs to be switched in and out of memory providing the ability to run different programs.

Unit 2 looks at only three machine code instructions and these were all given on the question paper in part (d) as a reminder to students. The correct answer only needed use of the LOAD and STORE instructions and over half of all students secured all three marks for this question part. A common mistake was to just see an answer of the form 'LOAD 21

ADD 22 STORE 23' showing that perhaps a student did not understand what the question was really asking them to perform which was swapping two stored values around.

Part (e) was a question looking at the differences between robotics and how we cope with situations. It was pleasing to see students identify aspects such as robots finding it hard to get feedback when gripping an item, and the problems in separating two similar coloured balls when they are obscuring each other. An answer that was rarely seen was the idea of a child building up experience over time and learning, compared to a robot just being programmed.

Students continue to struggle with identifying the major principles of how hardware devices work and it was common that no marks were achieved when discussing the digital camera and nearly 10% left this question part blank. Better answers considered items such as the shutter opening and closing to capture the image and correctly identifying the use of a sensor with more able students stating that this would be a CMOS or CCD device. It was surprising to see a few candidates talk about the use of film to capture the image.

The second part to the digital camera question was answered well and it was clear that students could link the idea of low resolution images to the needs of a robot. The simplest answer was to talk about the lower storage space required but the more able students linked this to the robot being able to process this amount of data faster or even that to identify colour differences would not require high resolution. Students should be encouraged to express their answers with direct reference to a question context, when appropriate, as this does allow them to demonstrate their understanding at a higher level.

Q11.

The majority of students scored at least one mark when explaining what was meant by the term bus. This was generally achieved by discussing how a bus connects components together inside a computer. It was not very common to see a good discussion about the use of parallel wires with many students just stating 'a wire', however students were still able to pick up full marks by providing other valid points. Weaker students talked about information being carried by a bus rather than signals and gave vague statements about carrying this 'back and forth'.

Part (b) proved to be harder for students to secure marks on and they struggled to provide a clear example of a control signal. Some students described situations that might cause an interrupt to be generated, for example, the pressing of a key on a keyboard, but usually these events were not themselves control signals. The common correct answers included a memory read or memory write signal with the more able students explaining this in further detail.

The diagram in (c) was answered well with the majority of students securing 3 or more marks. The directionality of the arrows for the address bus was the main cause for students to lose marks and it was surprising to see answers where no attempt was made to connect up the address bus. Students must be reminded to read questions carefully.

Of the students who did connect up the address bus a few did not realise that the keyboard and graphics controller would have a connection to the address bus.

Q12.

The question started off by asking students what an opcode and operand represent. Around half of students managed to gain the individual marks. A few students managed to get the answers around the wrong way. Students who answered by saying that the opcode was what the instruction was to do generally also gave an example and secured

the mark. Students who wrote only about the instruction need to remember that the instruction is both the opcode and the operand. A few students thought that the operand was where the instruction was carried out.

Compared with machine code a program written in assembly language is easier to understand and this was a popular answer. The students who stated that the assembly language program was easier to read were not awarded the mark as this was not considered to be a sufficient answer. It is clear that some students still struggle to understand the link between assembly code and machine code and how assembly code is translated. Students who wrote about compiling or compilation in their answers were not awarded marks. A few students stated that assembly code is executed faster than machine code and again did not gain any mark.

Q13.

This question, which also assessed QWC, was answered well by students with the more able securing a high score. Students generally appeared more able to discuss the fetch and decode parts of the cycle and often failed to describe the execute part. This limited their mark to a maximum of 5 out of the 6 available. The question did not place the list of registers in any order yet some students tried to follow the order provided and this therefore led to low marks.

A few students missed out the role of main memory completely with the contents of the MAR being passed straight to the MBR.

It was common for high scoring students to make many valid points beyond what was in the mark scheme and it was pleasing to see the depth of their knowledge. The role of the status register was, however, not well known and students thought that it either executed the instruction or actually controlled the fetch execute cycle. It was therefore pleasing to note when students wrote about overflow or underflow being indicated by a flag in the status register.

Q14.

The majority of students correctly identified that it was a third generation programming language for question part (a). Incorrect answers included students just writing ‘high level languages’ alongside the wrong answers of fourth and second generation.

The majority of students could also identify that the machine code was written in hexadecimal format. Students did struggle to provide a reason for using this format with ‘easier to understand’ being a common accepted answer. It does seem to be a common misunderstanding that hexadecimal uses less memory than binary / machine code.

Whilst hexadecimal memory addressing is included in the specification students struggled to identify the lowest and highest memory addresses that would be available. This was a question part that required the student to identify the op-code and operand part of the instruction and to know that hexadecimal characters range from 0 to F. Responses written in hexadecimal were appropriate; there is no requirement to convert from hexadecimal to denary in COMP2.

Asking students to provide a situation where it would be appropriate to use a low level language gave rise to lots of varied answers. Strong students identified programs that needed to be optimised for speed of execution or object code memory size. Whilst parts of an operating system might be written in a low level language, this was not awarded a mark on its own. However, students who identified device drivers or code used to directly manipulate hardware were awarded a mark.

The idea of compilers translating all in one go and interpreters translating and executing line by line is well known. Students also often secured a mark by stating that a compiler produces object code.
Marks were not awarded when students simply repeated parts of the question for example: ‘a compiler compiles...’ or, ‘an interpreter interprets....’. It was quite common to see, ‘an interpreter compile...’

Across the papers it was evident that a group of students were confused over the terms machine code, object code and source code. It was also evident that some students are confused over machine code, assembly code and high-level languages.

It should be noted that students need to be careful when writing about execution speed. It was common to see students write about compilers executing code fast. Students need to be aware that it is the machine code of the compiled program that executes faster compared with interpreting the same program.

Q15.

Question part (a) was answered well. A common mistake was for students to write MAR instead of Memory Address Register. Students need to make sure that they read questions carefully and write full names when requested.

Part (b) was less well known with weaker students just repeating back the question in forms such as, ‘it controls the processor.’ Students who described the control unit’s role in fetching, decoding and executing instructions picked up the mark.

Students seemed to struggle more than expected with part (c). Whilst a large number of students did identify the Arithmetic and Logic Unit, we saw a variety of responses using the initials A, L and U. The accumulator was also a common wrong answer.

Part (d) has been asked previously and we are looking for two points to appear in the answer. In terms of the processor, a register is a fast memory location located within the processor. A group of students could identify the memory location aspect of the response, but did not situate it within the processor and therefore could not gain any credit.

The status register is clearly mentioned in the specification, but students struggled to provide an example of when a bit within it might be set. Common incorrect answers were based around instructions completing or the computer being switched on or off. It was pleasing to see some students answer with overflow, underflow.

Q16.

Part (a) asked candidates to identify a segment of code as being from a second generation language. It was pleasing to see that most candidates could successfully identify this.

Part (b) asked candidates to identify a label as ‘memory address’. A variety of responses were given that did not secure the mark but ‘line number’ was accepted as the figure could have been showing a print out or part of a programming environment.

Questions relating to the terms opcode and operand have been asked before and the majority of candidates managed to secure full marks for part (c), but a reasonable number did not secure any marks.

Part (d) examined whether candidates realized the relationship between an assembly instruction and a machine code instruction. Candidates who indicated that there was a one to one relationship gained full credit. The majority of candidates compared the whole

of the code segment and gained the mark if they could identify the kind of language being used. It was pleasing to see strong candidates go further in their answer and understand that Figure 2 was the assembled version of the assembly code in Figure 1.

Q17.

An error appeared in the diagram. Following a review of a range of responses from candidates to part (a) by Senior Examiners and discussions with other AQA staff, the decision was taken that the most appropriate course of action would be to credit all candidates with the maximum 5 marks for part (a).

Parts (b) and (c) were answered well by the majority of candidates and they correctly gave the full name of the registers involved. Some candidates gave 'Current Instruction Register' for part (c) forgetting that the instruction would initially be loaded into the Memory Buffer Register.

Part (d) asked candidates to explain what is meant by 'a 64-bit address bus'. Weaker candidates answered with a rewording of the question by saying that a computer could move 64 bits at a time. A few candidates compared a 64-bit bus to a 32-bit bus which was not what the question was after. Strong candidates answered by correctly identifying that it would use 64 wires to transfer addresses of locations around the main components of the computer. Good answers also included that the computer would be able to address 2-64 memory locations whilst weaker candidates stated that the computer would be able to address only 64 locations.

Q18.

Part (a) was answered correctly by the majority of candidates, but some failed to secure the mark. Most of those who answered incorrectly gave the response ‘control bus’. The only unidirectional bus is the address bus.

Part (b) was poorly answered by the majority of candidates. Only a few candidates identified that one extra line would be needed to go from being able to access 4 GB of main memory to being able to access 8 GB. Candidates gave many varied answers with the most common being 32, 33 and 64. Whilst some candidates may have misread the question, it is clear that this topic was very poorly understood.

Part (c) was very well answered with the majority of candidates scoring highly. The most common mistake was to swap around processor and main memory; noting the direction of the address bus would have avoided this.

A few candidates used the answers ‘address bus’ and ‘data bus’ and it is clear that the bus system still causes confusion for a minority.

Q19.

In part (a) two thirds of candidates were able to identify the opcode and operand parts of the instruction correctly. Despite being given the instruction to split up, some candidates responded with different opcodes and operands.

Some responses used the binary number system; these were awarded credit if the number for the operand was correct and a translation table between opcode mnemonics and binary values was provided by the candidate.

Part (b) was answered correctly by the majority of candidates. Whilst some candidates added in extra instructions, these did not alter the final outcome and they were therefore awarded full marks. These extra instructions were generally to store a partial answer and

then to bring it back into the accumulator.

Q20.

This question covered another aspect of computer hardware, the fetch–execute cycle, and why programs are written in high level languages. The table was correctly completed in the majority of cases with the labelled parts of the processor. However some answers simply gave the acronyms rather than the full names of the registers, which the question had clearly asked for. Questions about the 'decode and execute' parts of the cycle have not been asked before this showed in the answers with many candidates describing the fetch part of the cycle and not what was asked. The part of the question concerning why programmers prefer instructions in hexadecimal compared to binary was often answered by saying it takes less storage space which clearly it does not. The answer to the question about the program translator was almost universally well known. When answering the final part, worth three marks, about why programs re written in a high level language, candidates often gave only two reasons and automatically failed to gain one mark. Answers stating that it is, 'like English,' were simply not enough and were marked accordingly; candidates needed to add to this to gain a mark.

Q21.

For part (a) the vast majority of candidates were able to name two registers that were involved in the fetch part of the fetch–execute cycle, and overall a wide range of the possible answers were given. A minority of candidates gave acronyms such as MAR and so failed to gain credit. Some candidates clearly knew the correct acronyms and tried unsuccessfully to invent or recall a corresponding full name, e.g. "Memory Bus Register", "Memory Access Register".

For part (b) candidates often attempted to describe what was happening in steps 1 and 2b without explaining the role of the two buses and main memory, which is what had been asked for. Very often candidates confused the roles of the data and address buses. There were many references to bidirectional data buses and unidirectional address buses but this did not answer the question that had been asked. Candidates often stated that the memory was moved rather than the "contents of" a specific location or its equivalent description. Also fairly common as a wrong answer was the fact that the address bus carried data from place to place.

Responses to part (c) were better than those to part (b). Nevertheless many candidates did not realise that the two processes are independent of each other. A very common wrong answer was that the buses are bi-directional and so allow two things to happen in opposite directions simultaneously. The idea that the steps 2a and 2b occur in different components was better known but often very vaguely stated. Candidates sometimes talked themselves out of a mark by stating the processes happen in different registers and different buses.

Q22.

Subpart (a) (i) asked candidates to identify the opcode and operand in an assembly language instruction. Surprisingly slightly less than half of candidates gave the correct answer. Despite being given the two options to choose from, some candidates wrote in different words and even binary numbers.

A register is a storage location that is inside the processor. For subpart (a) (ii), many candidates stated that it was a memory location but then did not go on to add "in the processor" or another equivalent correct statement. Answers often mentioned a storage location in the CPU. This is the final time the term CPU will be accepted as an alternative to processor. Traditionally, the term CPU refers to processor plus main memory so it is not

accurate to refer to a register as a storage location in the CPU. Vague answers such as “where data can be temporarily stored” were also common. A register was often confused with the operating system registry and answered accordingly.

Part (b) was on the whole well answered and, for able candidates, it was a very easy three marks. This was not a difficult question as candidates had been given an example in the stem of the question. However, many candidates gave from four to six program statements. Others also placed #s in the answers which was not required. Under this specification, candidates are not expected to be aware of the different addressing modes. Other candidates lost marks by giving the opcode two or even three operands.

Q23.

For question (a) the explanation of what is meant by a pixel was generally not well answered with very few candidates gaining the full 2 marks. The ‘smallest picture element’ was required for 2 marks to be awarded.

In question (b)(i) most candidates appreciated how the memory contents shown were arrived at from the grid of pixels given in the question. Some candidates did not read the rubric and gave the answer for question (b)(ii) in binary.

For question (c)(i) all that was required was a statement which described each colour being represented by a different number. Some candidates gave detail about numbers mapping to the various amounts of red, green and blue for each colour which was not expected, but was creditworthy.

In the final part of the question (d) despite being popular on the legacy CPT1 paper, answers describing vector graphics were disappointingly poor. Candidates failed to describe the two key points that any drawing is built up as a series of drawing objects and these drawing object types each have their own set of defined properties. Candidates were often unable to give a clear explanation for question (d), but were then able to name typical properties for a circle object.

Q24.

The best answers to this question were given by candidates who had learned the register transfer notation for the fetch-execute cycle and could therefore describe each step concisely and unambiguously. Some good answers were written as prose but candidates who took this approach often lost marks through either lack of understanding or inaccurate expression.

The most common misconceptions were in steps 2b and 4. During step 2b, the contents of the memory address stored in the MAR are fetched from the main memory into the MBR. The contents of the MAR are not simply transferred into the MBR. During step 4 it is the contents of the CIR that are decoded, not the CIR itself. As in January, many candidates attributed the CIR with the ability to do the decoding instead of realising that the CIR simply holds the instruction whilst it is decoded.

It was pleasing to see that many candidates understood that increasing the width of the data bus would enable more data to be transferred at one time. In contrast, only a small number of candidates understood that increasing the width of the address bus would increase the amount of addressable memory. Many mistakenly believed that it would allow multiple addresses to be sent simultaneously. Candidates appreciated that increasing the clock speed would speed the computer up, but many failed to explain that this would be because instructions could be executed more quickly so failed to gain credit.

Q25.

This question clearly showed which candidates had studied this topic. Answers mostly either gained full marks or hardly any. Candidates are not expected to be able to write lengthy low level code. They are expected to understand how a fairly simple assignment statement in a high level language would be represented in assembly code. Most candidates knew that an assembler is required to translate assembly code statements into machine code, but rather fewer candidates knew that only a compiler translates a high level language statement into machine code. An interpreter will not produce machine code; it merely interprets and executes high level language statements.

Q26.

This was a very poorly answered question, even though the Fetch-Execute cycle was given and candidates did not need to recall the exact sequence, only interpret the register-transfer language statements. Few candidates seemed to have any appreciation that a register is merely a memory location; it can't do anything except hold a binary value. Many candidates wrongly attributed the Current Instruction Register with the ability to decode and execute the instruction and the Program Counter with the ability to add 1 to itself. Some candidates were not at all sure what was stored in the Program Counter. Although the synonyms were given in the register–transfer language statements, candidates were expected to use the full names of the registers. Some candidates made up various fictitious names. Answers only using the abbreviated register names were credited this time, but in the future candidates may be required to give the full names.

Q27.

Parts (a) and (b) were generally well answered but less so (c) and (d). Many candidates gave a vague answer for the assembler output as 'errors' and this was considered insufficient.

Very few candidates were able to store the maximum 4 marks in part (d). There were very few convincing answers, and common misconceptions were that the program is executed from the hard disk, with no mention of program instructions. Often candidates generally described the need for the program to be loaded into main memory, without any further explanation.

Q28.

It would seem that many candidates did not have sufficient knowledge or understanding of assembly language. This question was badly answered with many candidates' responses suggesting that they had not been exposed to assembly language.

Q29.

In part (a) the most popular answers were 'processor' and 'RAM' but few candidates were able to give a convincing answer for the second mark, often merely re-stating the words in the question stem that the processor would 'increase the speed of execution of the program'. A good explanation for RAM was rare; candidates clearly have the knowledge that more RAM should result in the faster execution of a program but not the reason behind it.

The most popular answer for (b) was some form of hard drive with this time candidates usually able to get the second mark for a simple statement that it will, 'increase the storage capacity,' but note 'bigger disk' was not considered sufficient (a good example where clear quality of expression gets the mark).

For (b) 2 marks were rarely scored. Candidates who had used a USB on their computer would have identified with this question straight away, but an answer stating that the devices can be then 'simultaneously' connected was often missing.

Worryingly, as the content would seem to be at the heart of the subject at this level, there were a noticeable number of scripts which did not attempt some or all of this question. The distribution of candidate marks indicated that this question proved to be a good discriminator of candidates' ability.

Q30.

(a) Generally well answered.

(b) Generally well answered.

(c) Few candidates scored the full two marks. The idea that the data bus is used to transport data values typically from the processor to main memory, but also from the memory to the processor (hence requiring the bus to be bidirectional) - the address bus only ever transports addresses from the processor to the various devices; this was the level of answer expected.

(d) The specification says "arithmetic and logical operations". Credit was given to candidates who clearly had some practical experience of assembly language and so quoted particular types of operations from the instruction set. We stress that although practical experience of assembly language is not a CPT1 requirement candidates do need to understand that a processor performs basic machine operations.

(e) What was asked for was an understanding of the stored program concept and, despite this appearing on several previous CPT1 papers, answers given continually fail to describe this fundamental concept of how a digital computer works.

Q31.

A majority of candidates were able to describe the function of the Arithmetic Logic Unit and most of those were able to name at least two registers in part (c). Fewer candidates were able to describe the function of the control unit.

Q32.

Full marks were often achieved by candidates who had properly covered the fetch-execute cycle. A major source of error in identifying the components was due to a failure to use the fact that the bit pattern being retrieved from Main Memory was an instruction and should therefore arrive at the CIR.

Q33.

This was a very straightforward question but the responses were often disappointing. Parts (a) and (b) were generally done quite well but fewer candidates were able give a satisfactory response to part (c). Of those that did answer part (c) many failed to gain the second mark. This was another example of poor examination technique. Candidates should be encouraged to note the number of marks for a question and take this into account when deciding on their response.

Q34.

The majority of candidates were able to recognise (a) and (b) as assembly language and machine code respectively.

(a) Very few candidates were able to suggest the numbers represented memory addresses, despite previous CPT1 papers asking candidates to explain the stored program concept where the basic concept of a program resident in addressable memory is fundamental to any computer system.

(b) Candidates usually failed to get the mark by being too vague with explanations such as 'Fig 1 was the same as Fig 2', or 'did the same thing'.

The required answer was the fundamental relationship between assembly language and machine code; namely a one-to-one correspondence between the instructions.

(c) This was poorly answered.

Q35.

This question discriminated very well with the better candidates scoring high marks and most of the weaker ones managing to gain two or three marks at least.

Many candidates gained all four marks in part (a), while others lost one mark by referring to the registers by their initials only. Several inappropriate and some non-existent registers were named, while a few candidates did not attempt the question at all.

Part (b) was usually quite well answered, but a significant number of candidates where either inaccurate, or not clear enough, in their response to the second part.

Many good descriptions of the vectored interrupt mechanism were seen in part (c)(i), although a number of candidates did write about "executing the interrupt" rather than passing control to the start address of the ISR. Rather fewer candidates were able to suggest clearly why this mechanism makes the use of interrupts more flexible in part (ii).