7.3 Structure and role of the processor part 2

Q1.

(a) 1101 0001 0101 1010;;

1 mark for each correct byte
I leading bits

2

(b) 16;

1

[3]

Q2.

Need to access/address machine registers / exact memory addresses / hardware directly; fast speed of operation required // assembler code runs faster than programs written in HLL;
Code needs to take up little memory // assembler code gives smaller object code programs;
A minimise the size of the program/code;
No compiler/interpreter exists yet for machine // no other translator exists;
R manipulate bits

Max 2

[2]

Q3.

(a) From -2^{15}/-32,768; to 2^{15}-1/32767; **R** *binary;* **R** *one number unqualified*

2

(b) 2^{10}-1 / 1023; **A** 1111111111_2; **A** &3FF; **A** $3FF_{16}$;

1

(c) Step 1: (Content of) PC copied into MAR // [PC] → MAR; *accept {} for []*
Step 2: (Content of) PC incremented (by 1) // [PC] incremented (by 1);
Step 3: content of Memory location addressed by MAR; loaded into MDR;// MAR addresses a memory location; whose content is loaded into MDR;
Step 4: Content of MDR copied into CIR // [MDR] → CIR;
Step 5: Content of CIR decoded // [CIR] decoded // opcode/instruction/data decoded;
Step 6: instruction executed;

Step 2 could be after step 3,4 or step 5 but otherwise order of steps must be as above, step(s) may be missed out steps do not have to be one per line

P1 of it is difficult to pick out the steps from prose
A loaded/moved/stored for copied
A MBR for MDR
A IR for CIR
A SCR for PC

Max 6

[9]

Q4.

(a) The set / list of bit patterns / binary codes representing machine operations;
The set / list of bit patterns / binary codes for which machine operations have been defined;
The collection of different operations available;
A commands
R interpreted
R A set / collection etc

1

(b) 64 or 2^6

1

[2]

Q5.

(a) 1 – main memory;
2 – processor; **A** CPU;
3 – I/O port;
4 – address bus;
5 – control bus;
R anything else

5

(b) (i) (Processor) executes instructions; **R** data **R** programs **BoD** 'executes data and instructions';
(main memory) stores program/data currently in use;
A temporary storage of data/programs; **R** information **R** application
(secondary storage) holds programs/data/files for long-term/ non-volatile storage;
R application **I** virtual memory
A permanent storage of data/programs **R** information **R** backup

3

(ii) 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 examples *read/write on its own not enough*

Max 2

(iii) Instruction(s); address(es);

Max 1

[11]

Q6.

(i) A register / the accumulator;
A general purpose register
R the wrong register e.g. MDR

1

(ii) Access to main memory is slower than to a register;
Would need to write results to MM and read them back again for each instruction;

2

[3]

Q7.

A register:

(a) A storage unit where data / control information is temporarily stored /
A instructions
A storage unit which can be accessed rapidly /
A storage unit internal to the processor;
A storage unit that can be symbolically identified
A location for storage unit

Need the idea of storing

1

(b) Fast access / they have to be accessed frequently / execution faster;
Can have their own special instructions;
Can have a bigger word size;
No bus bottleneck / inter-register transfer along dedicated internal buses;

1

(c) *Any 3 from*

Accumulator	Holds the results of arithmetical calculations
MAR	Holds the address for data transfer e.g. read / write;
MDR / MBR	Holds data during transfer (between main memory and the processor);
PC / SCR	Holds the address of the next instruction (to be fetched / executed);
CIR	Holds the current instruction while it is being analysed and executed;
Status / Flag register	East bit is set or cleared depending on whether a certain condition holds;
Index	For use in index addressing
Interrupt	For vector mapping

Must have correct name
No mark for name alone.

3

[5]

Q8.

(a) Processor would have to be re-designed; (1)
Every time a new type of device was connected. (1)
OR
Voltages / signals required for correct operation of device; (1)
Different from voltages / signals used by processor.(1)
OR

So as not to slow the processor down; (1)
To cut down on the number of required ports;(1)
1 device controller can control more than one device of the same type;(1)

2

(b) *nb 'controller' or 'card' needed*
Floppy disc
Hard disc / IDE
Any serial device - mouse, printer, modem, Joystick
Any parallel device - printer, CD-ROM, CD-R, DVD, tape unit, zip drive, scanner.
SCSI - Zip drive, tape unit, CD-ROM, CD-R, DVD
USB
I/O controller
Sound card / MIDI interface card / graphics card
Network controller card
Any one
R keyboard, VDU

1

(c) *Any 2 points*
Address only goes (from the processor) to device controllers / main memory;
Regardless of whether the data is to be read from or written to that location;
No feedback / acknowledgement;
Any data transfer goes on data bus;

2

(d) *Any 2 × two each (1 for name, 1 for description)*
Memory Write: causes data on the data bus to be written to the addressed location
Memory Read: causes data from the addressed location to be placed on the data bus / MBR / MDR
I/O Write: causes data on the data bus to be output to the addressed I/O port
I/O Read: causes data from the addressed I/O port to be placed on the data bus
Transfer ACK: indicates that data have been accepted from or placed on the data bus
Bus Request: indicates that a component needs to gain control of the system bus
Bus Grant: indicates that a requesting component has been granted control of the system bus
Interrupt request: indicates that an interrupt is pending / transmission error
Interrupt ACK: acknowledges that a pending interrupt has been recognised
Clock / Timing: used to synchronise operations
Reset: initialises all components.
A combined Memory, I/O, Bus, Interrupt lines with suitable explanation

4

[9]

Q9.

(a) (i) (Data/address/control/internal/system) bus;
R just a description of a bus
R names of buses which don't exist e.g. memory bus

1

(ii) Store programs and/or data/files when not in use/
When computer is off permanent/long term storage

Of programs and/or data; save programs/data;
R offline/backup **R** ROM **R** temporary storage
A save on magnetic disk/ tape storage;
A information instead of data

1

(iii) (Machine code) instruction/data is fetched from main memory;
A what is fetched or from where
Instruction is decoded;
Instruction is executed (by the processor); **R** data executed

Max 2

(b) (i) Assembly language; mnemonic code; mnemonics; assembly code;
R low level language **A** assembler;

1

(ii) Translated/assembled/converted/decoded; into machine code (instructions);
R compiled **R** interpreted **A** object/target code;
A binary instructions;

2

(iii) Computer executes instructions in <u>programmer</u> defined sequence;
A the programmer tells the computer how to do it;
R user *instead of* programmer

1

(iv) Pascal /Visual Basic/Basic/C/C++/Cobol/Fortran/Ada/Delphi/Lylix/Modula /or any other imperative HLL
R Prolog
R Lisp
R Pop11

1

(v) One statement/instruction/command in a high level language translates into several machine code instructions; 1 to many;

1

(vi) Laborious/time-consuming to write; hard to debug; harder to program; easier to make mistakes; more difficult to understand/ learn; difficult to maintain; different assembler/instruction set for different type of computer; machine dependent; low level programs not portable;

Max 2

[12]

Q10.

(a) Group/number of bits which can be addressed/transferred/manipulated <u>as a single unit/in one go/at a time</u>;
R processed instead of manipulated
R amount of data instead of number of bits

1

(b) <u>0 1</u> ; *(must have both values and only those values)*
R on or off on its own

1

(c) Instruction/part of program/machine code; opcode & operand;

(pure binary) integer;
Real/floating point number/fixed point number; **R** number **R** binary
R hexadecimal
(Unicode/EBCDIC/ASCII) characters; **A** text;
Pixel(s); **R** bitmap **A** part of a bitmap boolean; a set of flags;
BCD;
Address/pointer;
A operand;
A sound;

Max 3

[5]

Q11.

(a) Clock speed
The frequency at which a clock pulse occurs;
R pulse/cycle period.

1

(b) **R** frequency of fetch execute cycle
Increasing the clock speed will increase the speed at which instructions are executed.

1

[2]

Q12.

Any 3 from:
PC/SCR - Address of next instruction; - To be fetched;
MAR - Address of data &/or instructions; - That are to be read from - (main) memory;
MBR / MDR - Data &/or instruction; - Read from (main) - memory;
IR/CIR - Holds current instruction; - While it is being - decoded and executed;

Correct name/acronym required. Accept other correct 'Further details' – but must refer to role of register being described. May be incorporated in 'what stored' and carried forward.

[9]

Q13.

(a) System software / program which controls the computer hardware;
Manages computer (system) / hardware;
Interface between user and computer;
Runs programs; handles input/output;

Max 1

(b) (i) Executes instructions/programs/code; **R** data
Performs calculations/instructions
R controls…
R processes…

1

(ii) Stores/holds program /instructions / data;
R permanently
R files

1

(c) **Data bus**; (1)
carries the data/instructions to/from component; **R** holds Address bus; (1)
carries identification/address about where the data is being sent to /fetched from; **R** holds (1)
Control bus; (1)
to send control signals; whether process is read or write; (1)
carries timing signal;
R holds
R controls flow of data

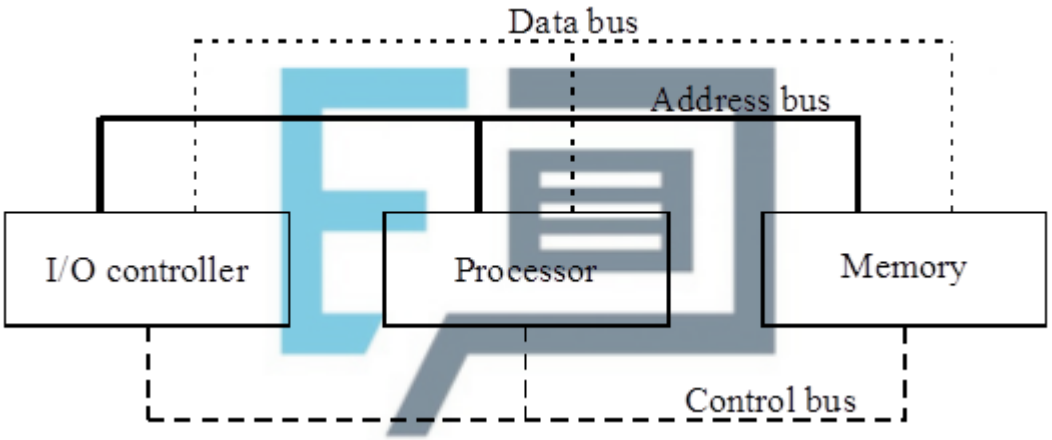
1 mark for name 1 mark for example × 3

6

[9]

Q14.

Bi-directional double arrow anywhere on bus with no contradictory arrows
Address bus must be clearly arrows



(a) Address bus arrows(1)
Data bus arrows(1)
Control bus arrows(1)

3

(b) Characters / ASCII; ***(NOT** text)*
Integers; (not numbers, binary)
Real numbers / floating point numbers / fixed point numbers;
BCD;
Instructions, (machine code);
Bitmaps / vector graphics / encoded picture;
Encoded sound;
An address;
Status information;

1 mark for any two different types

Max 2

(c) Signals;
Status information;

1 mark for one type

Max 1

[6]

Q15.

Data Bus
Address Bus
Control Bus
Or any other appropriate bus

Any 3 - 1 for name, 1 for description

[6]

Q16.

(a) Machine code
Instructions that a computer can actually execute
Coded in binary
Machine dependant
Assembler language
Uses mnemonics
Version of machine code / **1 – 1 correspondence**
Easier to understand than m/c code
Machine dependant

For each, 1 mark for name, 1 for up to two descriptors

Max 6

(b) Assembly language enables close manipulating of bits / bytes etc. / **device drivers**
Executes very quickly
Uses less memory than a comparable HLL version
To maintain historic coding

1 mark for a reason

Max 1

[7]

Q17.

One from: PC/SCR, MAR, MBR(MDR). LR/CIR(not address/data register)

[1]

Examiner reports

Q1.

(a) Most candidates gained full marks for converting the hexadecimal digits to binary.

(b) However, many candidates could not see that the address bus would require 16 lines to convey the resulting 16-bit number.

Q2.

Many candidates gave very imprecise answers here, many claiming that it was easier to write in assembly code. Some candidates compared writing in assembly code with writing in machine code, missing the point of the question. The required answer was that writing in assembly code can produce more concise code and so the code will run faster and take up less memory.

Q3.

(a) It is very disappointing to see so few candidates give the correct answer to this question. When 2's complement is used, the possible range of integers is from -2^{15} to $2^{15}-1$ or 32,768 to +32,767 for a 16-bit word.

(b) Although the number of bits available for the operand was given as 10 in the question, few candidates deduced from this that the highest possible address therefore could only be 1111111111_2 or $2^{10}-1 = 1023$

(c) Many candidates had a fair understanding of the fetch-execute cycle but could not explain in logical steps what was happening. Since the PC points to the next instruction to be executed, the steps are:

1. copy the address held in the PC into the MAR.
2. the content of the memory location addressed by the Mar is loaded into the MDR
3. the content of the MDR is copied into the CIR
4. content of the CIR is decoded
5. the instruction in the CIR is executed

the PC is incremented either after step 1, 2, 3 or 4 but definitely before step 5.

Q4.

Although many candidates scored well on this question, others showed a basic lack of understanding of this topic area. For example, part (b) asked 'With 6 bits of the op code reserved to denote basic machine operations, how many basic machine operations may be coded?' Incorrect answers included 1, 2, 6, 13, and 63.

Q5.

(a) This being almost a multiple-choice question meant that nearly all candidates picked up some marks. The majority of candidates did not pick up on the significance of the uni-direction arrow from the processor to the address bus as the

major clue.

(b) This part was very poorly answered with the usual 'processor processes data' being a typical response in (i). Students who have knowledge of von Neumann architecture know that instructions have to be in main memory at the time of the fetch-execute cycle, though others could have picked up the mark by stating that the main memory is a temporary store for programs/data. Many candidates lost even that possibility by stating that it stored 'information'. In Computing 'information' is not a synonym for 'data'.

Part (ii) saw hardly any credible answers. Interrupts, clock signals and memory read, memory write were the usual correct answers.

Part (iii) saw a few correct answers with some losing the mark by stating the too vague 'location' rather than the more specific address, or commands rather than instructions.

Q6.

Most candidates knew that intermediate results were stored in a register, though many chose the MDR, MDB, CIR or status registers instead of the accumulator. A general-purpose register was also accepted. RAM, cache memory and a stack were not credited.

Many candidates had a vague idea that access was faster to and from the registers than to and from main memory, but few managed to get the second mark by expanding on this answer. Credit was given for a good explanation as to why this was so. The fact that this was an intermediate result, and so more calculations were to be carried out on this data, also seemed to have been overlooked by many candidates.

Q7.

This focused on registers. This was a bookwork question, and careful candidates gained full marks. However, although many candidates knew how registers were used in the standard fetch-execute cycle, fewer could give a generic definition of a register. Many could name three, and more, individual registers, but fewer could give a good explanation of their purpose. Marks were lost through careless terminology, e.g. 'MDR; this is where data is written to or read from memory', 'PC; this shows the current stage the program is at (the next instruction to be executed)'. The accumulator does not carry out most calculations; it stores the intermediate values and final results from them.

Q8.

This focused on device controllers and buses. Candidates did not seem to appreciate that the voltages required for the correct operation of devices were different from those required by the processor. Thus, without a device controller, the processor would have to be re-designed every time a new device was installed. The use of device controllers can also cut down on the number of ports required as one controller can control more than one device of the same type.

The reason why the address bus only carries addresses in one direction is that the processor has to send the required address to the controller or to main memory for both the read and write operations. No data or signals have to return via the address bus from the device to the processor. The knowledge of typical signal lines in the control bus was patchy.

Q9.

(a) (i) Most candidates correctly named one or more buses. The term 'bus' was enough to gain the mark, but some candidates still referred to a 'memory bus' which did not gain credit.

(ii) Very few candidates seem to appreciate that secondary storage is used to save programs and data when they are not in use. Most referred to backup copies, which may well be saved on secondary storage, but is not the primary purpose of such storage.

(ii) The fetch-execute cycle was very well explained by a few candidates, though in too much detail by some others (e.g. by those who had presumably just studied machine architecture for CPT4). The majority of answers involved fetching data from memory and then executing data which gained no credit. Correct responses stated that an instruction is fetched from main memory, decoded and executed by the processor. At this machine level of operation, the term 'information' is not appropriate.

(b) (i) Assembly languages are second generation programming languages. The term 'assembler' was accepted this time, but candidates should be able to distinguish between the two terms and appreciate that the assembler is the translator, which converts the assembly language program into machine code.

(ii) Many candidates lost a mark by wrongly stating that a compiler or interpreter converts an assembly language program into machine code. The terms 'source code' and 'object code' belong to the translation of high level language programs by compilers and should not be used in the context of second generation languages.

(iii) Very few candidates could explain what the term 'imperative' meant in the context of high level languages. Most thought it meant important or problem oriented. Of those who were on the right track, some then confused the definitions of imperative and declarative languages. A correct response explained that the computer executes instructions in programmer-defined sequence. It was not acceptable to equate a programmer with a user.

(iv) A great many different languages quoted here gained credit. However, Prolog or HTML were not acceptable examples.

(v) Few candidates could state that one high level language statement would translate into one or more machine code instructions. Some candidates denied that there was any relationship.

(vi) This was a well answered question even for middle-scoring candidates, though the answers were sometimes a little vague. 'Hard to learn' and 'debug' were probably the most common answers which gained credit.

Q10.

(a) This question was not very well answered by the majority of candidates. A correct response was that a word is the number of bits which can be addressed or transferred as a single unit. Candidates need to understand that this is not necessarily the same as the number of bits which can be processed at the same time.

(b) Surprisingly many answers were NOT 0 and 1 but 1/8th of a byte or 1, 2, 4,8 16 etc.

(c) This question did not seem to be well understood by the majority of candidates even though a similar question appeared in January 2001 for CPT1. Candidates scored full marks if they appreciated that a 32-bit word could take 4 ASCII characters or 2 UNICODE characters, an integer or a real, an instruction or an address. 32 bits are not enough to store a bit-map, so candidates were expected to suggest part of a bit-map or pixel(s). A few candidates noticed that 32 bits was just enough to take an IP address.

Q11.

(a) All that candidates were asked to do in this part was to define clock speed as 'the frequency at which a clock pulse occurs.' Many did not recognise the system clock as a distinct component and used the term 'processor speed'. Some had difficulty distinguishing between the frequency of the clock pulses and the speed of the pulses. Others explained factors which depended on the clock speed, such as the rate at which the fetch-execute cycle was executed. A few good answers described electronic pulses emanating from a crystal and related the clock speed to their frequency.

(b) Here the candidates were asked to say what effect increasing (or decreasing) clock speed will have on the speed at which instructions are executed. Again, it is important to note that this question did not ask about how fast the processor worked or the effect on the cycle.

Q12.

Clearly, some candidates lost marks on this question because they did not read it properly. It asked about registers which were involved in the fetch part of the fetch execute cycle. Most answers correctly identified three registers, but a common fault was in terminology. Candidates used the word 'location' for a Memory cell, for the address of a Memory cell and for the contents of a Memory cell. It was sometimes apparent from their remaining answers what was meant, but in many cases it was not and was sometimes contradictory. Another example of laxity was in ascribing an activity to a passive storage device such as 'The Program Counter passes the address to the Memory Address Register'.

Q13.

(a) The idea that an operating system was an interface or that it controlled the computer was well known.

(b) The functions of a processor and main memory were often answered in too general terms. Too many candidates thought the processor was in overall control or giving timing signals. At this basic component level, candidates need to appreciate that the processor executes instructions, such as logical or arithmetic operations on data (not information). Main memory is volatile and stores the instructions and data of the programs currently running. Many candidates confused main memory with backing store and implied that data and software are stored there permanently.

(c) Most candidates correctly named the address, data and control buses. Far fewer could express clearly enough their use. Many answers attributed too much intelligence to a bus. "The address bus decides where the data should go" is not an appropriate answer. Many candidates imagined the bus to be moving carrying data and the control bus acting as a control rather like a set of traffic lights to stop other buses colliding. Good answers could explain that the address bus carries the address of where the data is being sent to or fetched from; that the data bus transfers the data between main memory and the processor; and that the control

bus transfers control signals such as whether the memory access is read or write and the timing signal.

Q14.

This is clearly one topic area that is not taught in all centres. Many candidates drew contradictory arrows. In part (b), many candidates failed to suggest interpretations of the data and instead gave its source and destination. More candidates knew that signals and status information are carried by the control bus.

Q15.

Candidates tended to either do very well in this question or else they were unable to answer this question at all. Many candidates were aware of the data bus and the address bus and the functions they perform. Fewer candidates were clear as to the purpose of the control bus.

Q16.

The examiners were interested to learn that COBOL, PASCAL, FORTRAN, and C++ were all low level programming languages, along with Imperative and Declarative. Those candidates, who did understand what low level languages were, gained most of the available marks, and gave good valid answers to part (b).

Q17.

Many candidates successfully answered with PC/SCR, MAR, MBR (MDR), IR/CIR abbreviated or otherwise for the name of one register involved in the fetch part of the fetch-execute cycle. Address register and data register were not acceptable answers. A few candidates stated, incorrectly, status register and accumulator.