Mark schemes

**Q1.**

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]***

**Q2.**

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

|  |  |
| --- | --- |
| **Quantity** | **Position** |
| 3 kilobytes | 3 |
| 2 mebibytes | 5 |
| 2 bytes | 1 |
| 2 megabytes | 4 |
| 20 bits | 2 |

**Mark as follows:**

**1 mark** for bits, bytes and kilobytes in correct positions

**1 mark** for mebibytes and megabytes in correct positions

**2**

(b)  **Marks are for AO2 (apply)**

**1 mark** for correct conversions between representations, allowing follow through for final answer.

2716 = 0010 01112

C916 = 1100 10012

Final answer: F016

**1 mark** for binary addition 111100002 allowing follow through if conversion was incorrect.

**2**

**[4]**

**Q3.**

**All marks AO1 (understanding)**

|  |  |  |
| --- | --- | --- |
| **Level** | **Description** | **Mark Range** |
| 4 | Description covers all, or almost all, of the points in the indicative guidance and fully reflects the sequence in which steps occur. It includes use of registers, buses and main memory. An excellent level of understanding is shown with no misconceptions. | 4 |
| 3 | Description covers most (ie more than half) of the points in the indicative guidance and completely or almost completely reflects the correct sequence in which steps occur. At least two of the use of registers, buses and main memory are covered. A good level of understanding is shown. Whilst there may be some omissions, there is at most one misconception in the response. | 3 |
| 2 | At least two correct points are made from the indicative guidance and there is some indication of understanding of the correct sequence. Some understanding is shown. | 2 |
| 1 | At least one relevant point has been made. There is not sufficient evidence to conclude that the cycle has been understood. | 1 |

**Guidance – Indicative Response**

•   Contents of Program Counter/PC transferred to Memory Address Register/MAR

•   Address bus used to transfer this address to main memory

•   Fetched value/instruction transferred using the data bus

•   Contents of addressed memory location loaded into the Memory Buffer Register/MBR

•   Transfer content of Memory Buffer Register/MBR to the Current Instruction Register/CIR

**A**. Memory Data Register / MDR for MBR

**I**. Incrementing of program counter, even if incorrect

**NE**. Points made using register transfer notation only eg CIR ← [MBR]

**[4]**

**Q4.**

**4 marks for AO3 (programming)**

**Example 1:**

          LDR R0, 100

          LDR R1, 101

          ADD R2, R0, R1

          CMP R2, #26

          BLT store

          SUB R2, R2, #26

store:    STR R2, 102

**Example 2:**

          LDR R0, 100

          LDR R1, 101

          ADD R2, R0, R1

          CMP R2, #25

          BGT adjust

          STR R2, 102

          HALT

adjust:   SUB R2, R2, #26

          STR R2, 102

**Example 3:**

          LDR R0, 100

          LDR R1, 101

          ADD R2, R0, R1

          CMP R2, #25

          BGT adjust

          B end

adjust:   SUB R2, R2, #26

end:      STR R2, 102

**A**. Use of alternative registers

**A**. Any label name in place of store / adjust

**DPT.** Use of invalid register name eg Rd

**DPT.** Use of incorrect addressing mode

**DPT.** Inclusion of invalid symbols in commands

**Programming Marks:**

**1 Mark** for LDR R0, 100, LDR R1, 101 and STR R2, 102

**1 Mark** for ADD R2, R0, R1

**1 Mark** for SUB R2, R2, #26

**1 Mark** for either:

•   CMP R2, #26, BLT store and store: aligned to a STR instruction **or**

•   CMP R2, #25, BGT adjust and adjust: aligned to a SUB instruction

**Max 3** if any errors.

**[4]**

**Q5.**

(a)  **All marks AO2 (analyse)**

**One mark** for correct purposes given for one or two registers **OR two marks** for correct purposes given for all three registers.

|  |  |
| --- | --- |
| **Register** | **Purpose** |
| R1 | The plaintext letter // the letter before it has been encrypted // the original letter // the characterCode |
| R2 | The key // the number of positions to shift letters by // the value to add to the letter // the keyValue |
| R3 | The ciphertext letter // the encrypted letter // the encryptedCode |

**A.** “letter” for “character” and vice-versa

**2**

(b)  **Mark is AO3 (programming)**

MOV R3, R1;

**I.** missing comma

**A.** another command which would achieve the same affect eg ORR R3, R1, #0

**1**

(c)  **All marks AO3 (programming)**

R3 compared to 90 or 91 as first command;

26 subtracted from R3 and result stored back into R3 after comparison (whether comparison is correct or not);

Fully working code;

**DPT** omission of # for immediate addressing values

**DPT** use of register number other than R3

**I.** missing commas

**Example Solution 1**

  CMP R3, #91

  BLT finished

  SUB R3, R3, #26

  B finished

**Example Solution 2**

  CMP R3, #90

  BGT moveBack

  B finished

moveBack:

  SUB R3, R3, #26

  B finished

**Example Solution 3**

  CMP R3, #90

  BLT finished

  BEQ finished

  SUB R3, R3, #26

  B finished

**3**

**[6]**

**Q6.**

(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; |  | Both Add instructions for the mark - do not |
| ADD   3 | need to follow each other. |
| STORE  | 21; |   |

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]**

**Q7.**

**Marks are for AO1 (understanding)**

**Purpose of start bit (Max 1)**

Start the receiver clock (ticking);

**A.** To wake up the receiver.

Synchronise / bring into phase the clocks of the transmitter and the receiver;

**A.** To synchronise the receiver and transmitter clocks.

**NE.** Synchronise the (two) clocks.

**Purpose of stop bit (Max 1)**

Allows the next start bit to be recognised;

Provides time for the receiver to process / transfer the received data;

**A.** Allows received data to be processed.

**NE.** Signals received data can be processed.

**[2]**