# Worksheet 3 Programming language classification Answers

**Task 1**

The following table shows some of the instructions available in an imaginary computer which uses 4 bits for the opcode and 4 bits for the operand.

|  |  |
| --- | --- |
| **Instruction** | **Meaning** |
| 0000 | Load the value stored in memory location specified by the operand into the accumulator |
| 0001 | Store the value in the accumulator in memory location specified by the operand  |
| 0010 | Add the value specified in the operand to the value in the accumulator |
| 0011 | Compare the contents of the accumulator with the contents of the location specified by the operand |
| 0100 | Jump to the address held in the operand if the accumulator held the lesser value in the last comparison |
| 0101 | Jump to the address held in the operand if the accumulator held the greater value in the last comparison |
| 0110 | Jump to the address held in the operand |
| 1000 | Stop |

1. (a) Add comments to each of the following instructions to say what it does.

Memory instruction

address

 1 0000 1001 ; load value from memory address 9 into the accumulator

 2 0011 1010 ; compare with value in memory address 10

 3 0100 0110 ; smaller value in accumulator so branch to instr. at 6

 4 0001 1011 ; larger or equal value in 9 so store in address 11

 5 0110 1000 ; branch to instruction held in address 8

 6 0000 1010 ; load value from memory address 10 into the accumulator

 7 0001 1011 ; store accumulator value (from 10) in address 11

 8 1000 0000 ; stop

 9 0000 0110 ; binary value 6

 10 0000 0111 ; binary value 7

(b) What will be stored in memory address 11 if memory address 9 holds 6 and memory address 10 holds 7? 7

 What if memory address 10 holds 5? It will then hold 6, the value at address 9.

(c) What is the purpose of the program, in general terms?

 It stores the larger of two values (or the value from memory address 9 if they are equal) in location 11 (binary 1011).

2. Write a machine code program to add the values in memory addresses 12, 13 and 14. Store the result in location 15.

 0000 1100 ;load value from memory address 12

 0010 1101 ;add value from memory address 13

 0010 1110 ;add value from memory address 14

 0001 1111 ;store result in 15

**Task 2**

Some assembly code instructions for a processor are given in the table below.

|  |  |
| --- | --- |
| **Instruction** | **Meaning** |
| LDA | Load the value stored in memory location specified by the operand into the accumulator |
| STO | Store the value in the accumulator in memory location specified by the operand  |
| ADD | Add the value specified in the operand to the value in the accumulator |
| CMP | Compare the contents of the accumulator with the contents of the location specified by the operand |
| BLT | Jump to the address held in the operand if the accumulator held the lesser value in the last comparison |
| BGT | Jump to the address held in the operand if the accumulator held the greater value in the last comparison |
| JMP | Jump to the address held in the operand |
| STOP | Stop |

1. Rewrite the program in Task 1, Question 1 in Assembly language.

 1 LDA 9 ; load value from memory address 9 into the accumulator

 2 CMP 10 ; compare with value in memory address 10

 3 BLT 6 ; branch if less than value in address 10 to instr. at 6

 4 STO 11 ; larger value in 9 so store in address 11

 5 JMP 8 ; branch to instruction held in address 8

 6 LDA 10 ; load value from memory address 10 into the accumulator

 7 STO 11 ; store accumulator value (from 10) in address 11

 8 STOP ; stop

 9 0000 0110 ; binary value 6

 10 0000 0111 ; binary value 7

2. Write a pseudocode algorithm that will perform the same task as the machine code program in Task 1 Question 1.

 IF a > b THEN

 max = a

 ELSE

 max = b

 ENDIF

3. What are the **disadvantages** of high-level programming languages compared with low-level languages?

* The object code (compiled or interpreted code) may run slower than assembly code or machine code
* The object code may occupy more space in RAM – which can be a problem in embedded systems with a small amount of memory
* Most high-level languages do not have statements to allow the programmer to manipulate individual bits – essential in some applications, e.g. device drivers

(Answers given on slide)