**ANSWER**

**DIFFERENCES**

|  |  |
| --- | --- |
| **High-Level Languages** | **Low-Level Languages** |
| The high-level language is programmer friendly. | Low-level language is machine amiable which means it is interpreted by machines easily. |
| When it comes to memory consumption the High-level languages are highly inefficient | When it comes to memory consumption the low-level languages are highly efficient |
| A high-level language is easily understandable | Low-level language cannot be interpreted easily as it contains a set of long series of 0’s and 1’s. |
| The programs written in high-level languages are portable and machine independent. | Low-level languages cannot run over different machines as these are not- portable and machine independent.  |
| Debugging and maintenance are easier | Debugging and maintenance are harder |

|  |  |
| --- | --- |
| **High-Level Languages** | **Assembly Languages** |
| In high-level language programs run independently of processor type. | In assembly language programs written for one processor will not run on another type of processor. |
| Performance and accuracy of high-level language code are not as good as that of Assembly language.  | Performance and accuracy of assembly language code are better than a high-level. |
| The compiler is used to compile code in the high-level. | Assembler is used to translate code in assembly language. |
| The executable code of high-level language is larger than assembly language code so it takes a longer time to execute. | The executable code of assemblers are much smaller than that of High-level Languages so it takes a shorter time to execute. |
| In assembly language, we can directly read pointers at a physical address which is not possible in high-level | In assembly language, we can directly read pointers at a physical address |

|  |  |
| --- | --- |
| **Low-Level Languages** | **Assembly Languages** |
| Machine language is in form of 0’s and 1’s. One showcases the true/on while zero depicts the false/off state. | Assembly language is English syntaxes, which is understood by CPU after converting it to low level language by interpreter and compilers. |
| CPU can directly understand Machine language, no need of compiler or assembler. | Programmers can understand the assembly lang.uage, however CPU cannot |
| Machine code differs platform to platform | Assembly language is set of instructions which in same irrespective of platform |
| Binary codes cannot be memorized. | The codes and instructions of assembly languages can be memorized |
| Modification is not possible. It has to be written from scratch for specific type of CPU. | Modification is not that tough here. |

**SIMILARITIES**

1. They all have rules and languages
2. They are all programming languages.
3. Their end result after they have been compiled/interpreted/assembled is machine code.
4. They are used to write computer programs.
5. They are advancements of each other.