2.1 Describe the process of translating a program written in a High-Level Language (HLL), e.g. C++, into an executable file that is ready for execution. Identify the system programs used in the process and describe the role of each of them assuming that the compiler generates an assembly language file as the output.

What is the system program used by the Operation System (OS) to load an executable file to memory and run it?

ANSWER

To translate a program written in a high-level language into simple instructions that hardware can understand, the high-level operations shall go through layers of software each interpreting or translating them into simple instructions.

The traversal of converting a high-level language program into simple instructions involves three main steps and involves several system programs.

The first step of translation involves a system program called the **compiler** which converts source code to assembly language (i,e a symbolic representation of the binary language(machine language) understood by the underlying hardware)

Assembly language code generated shall be translated into an object file by the system program **assembler.**

Object files of different modules are combined together to form an executable file. This task of linking object files is performed by the system program called **linker.**

A **loader** is part of the operating system that reads the executable file and loads those instructions into memory along with preparatory tasks to run the executable.

2.2 Describe the elements, including optional ones, of a MIPS assembly language statement.

ANSWER

MIPS assembly language instructions have the following format :

[label:] operation [operand1 [operand2 [operand3]]] [#comment]

LABEL(optional)

They are typically associated with a memory address.

OPERATIONS :

Native instructions

They have a one to one relation with machine code.
Eg: add, sub, lw etc

Pseudo instructions

They are added to simplify a programmer's task. These instructions are translated into one or more native instructions by the assembler
Eg: move, li, bgt etc

Assembler directives

These are not part of generated object files. They help assembler in defining segments and allocation of memory for variables
Eg: .data, .text

OPERAND:

Operands can be of the following types :

1. Register names
2. Immediate values
3. Address label

COMMENT:

They are used to express the functioning of a program. Typically start with #.
They are ignored by the assembler during translation of assembly language to an object file.