Research 1 of fundamental algorithms.

1.1 Stages of program compilation

What is Program Compilation?

Program compilation is defined as the process of translating a high-level program into a low-level machine where the code is in binary using the copiler. The computer, due to its hardware architecture, can only execute instructions in binary code. And therefore, all programs written in any lauguage other than the machine code must be firts converted to machine instructions.

Program compilation has several phases that convert the high-level computer program into human-readable, low-level code to be understood by the machine in binary format.

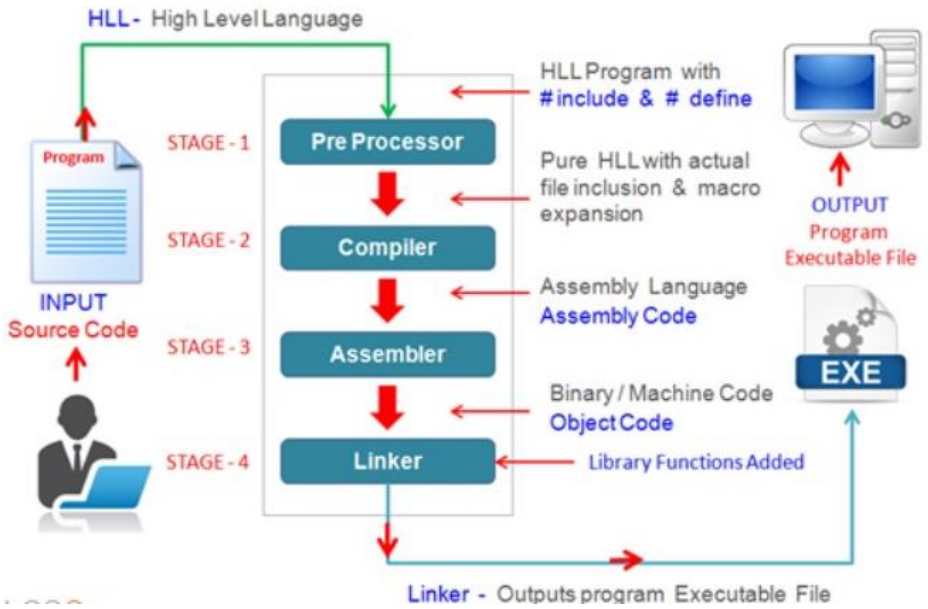

Image 01. The conversion of program source code into executable file (dot exe) happens in four stages. Thes four stages of compilations are pre-processing, compiler, assembly and finally the linking stage.

1.2 ***Program Compilation Process***

Four Stages of Program Copilation Explained

A compiler is a special type of system software is used to translate the program source code file into machine code. The machine code consist of machine instructions in binary.

The machine instructions are encoded by compiler in specific format called instruction format. The CPU decodes the machine instructions as directed by the instruction format.

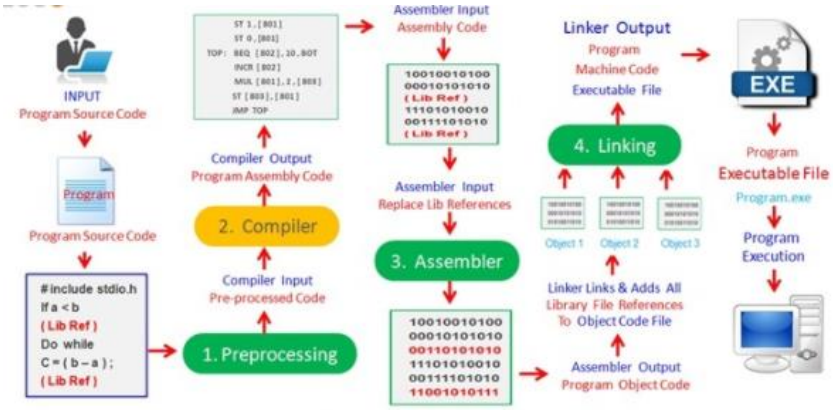

Image 02. The high level computer programs are compiled in four stages. Each stage of the compilation process takes the input from previous stage.

Stage 1 – Preprocessing

The first stage of the compilation process is called the pre-processor stage. Alternately, this stage is also referred as lexical analysis stage. The stage takes the program source code file as input and provides pre-processed file as output with dot i extension.

The pre-processing stage includes all the header files, all macros are resolved by replacing them with absolute values and the comments are excluded.

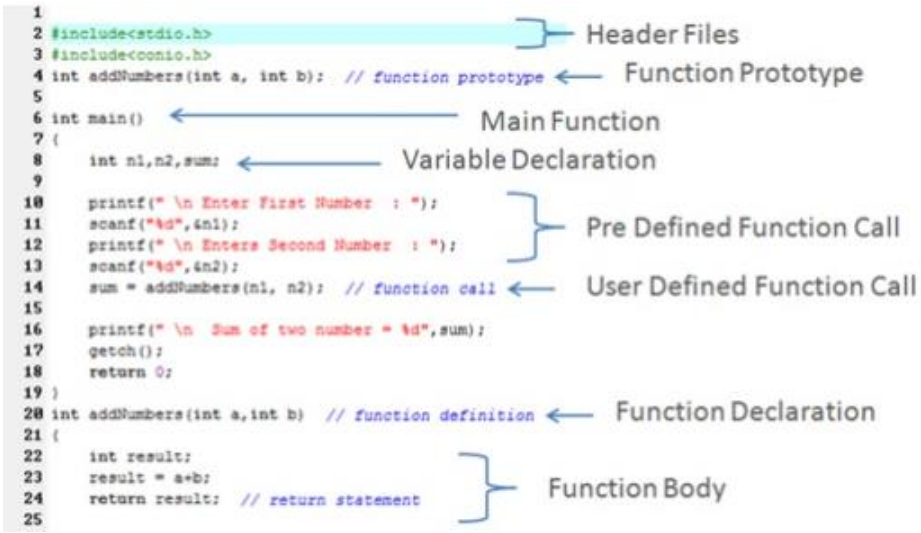

Image 03. In C: programming, the standard library functions are stored into separate header file with dot h extension. The separated header file helps to better organize the program code and improve readability.

Stage 2 – Compilation

The compilation is the second stage of the program compilation process. The compiler accepts the pre-processed file as input and provides the assembly code as output file with dot s extension.

```
1
2  // www.learncomputerscienceonline.com
3  #include<stdio.h>
4  #include<conio.h>
5  int addNumbers(int a, int b);  // function prototype
6
7  int main()
8  {
9      int n1,n2,sum;
10
11     printf(" \n Enter First Number  : ");
12     scanf("%d",&n1);
13     printf(" \n Enter a Second Number  : ");
14     scanf("%d",&n2);
15     sum = addNumbers(n1, n2);  // function call
16
17     printf(" \n  Sum of two number = %d",sum);
18     getch();
19     return 0;
20 }
21 int addNumbers(int a,int b)  // function definition
22 {
23     int result;
24     result = a+b;
25     return result;  // return statement
26
27 }
```

Image 04. The compiler converts all high level program instructions into its equivalent assembly code instructions. These instructions are platform dependent and compiled for a specific architecture.

Stage 3 – Assembling

The compiler accepts the pre-processed file as input and provides the assembly code as Output file with dot s extension.

```
        ST 1,[801]
        ST 0,[802]
TOP:    BEQ [802],10,BOT
        INCR [802]
        MUL [801],2,[803]
        ST [803],[801]
        JMP   TOP
BOT:    LD A,[801]
        CALL PRINT
```

Image 05. The compiler converts all high level program instructions into its equivalent assembly code instructions. These instructions are platform dependent and compiled for a specific architecture.

Stage 4 – Linking

The linking is the final and fourth stage of the compilation process. The main function of the linking is to produce a single executable file (Sourcefile.exe) by linking all the object code files.

The large Computer Programs are organized into number of manageable files. The user defined functions are written in a separate file. These files are tied to the main program file in the header section such as # include in C: language.

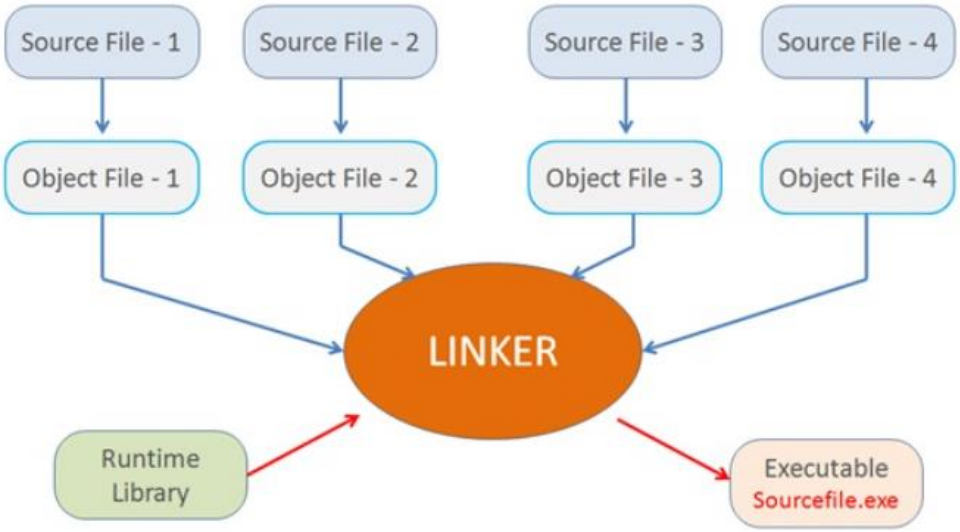

Image 06. The programmer can use these user defined functions and standard libraries by simply including these files in the beginning of the program header section.

2.1 Levels of programming

These languages vary in the level of abstraction they provide from the hardware. Some programming languages provide less or no abstraction while some provide higher abstraction. Based on the levels of abstraction, they can be classified into two categories: Low-level language. High-level language.

I. Microcode
   - Machine-specific code that directs the individual components of a CPU's data-path to perform small-scale operations.
   - People who build computers program in micro-code. The programs that you write are converted (as explained later) into machine code.
   - Every machine code instruction tells the CPU to execute a certain microprogram, written in micro-code.
   - Often these programs are implemented in hardware.

   Some microprocessors examples are:
   Many digital signal processing chips that mobile telephones use, FPGAs, or reconfigurable they can actually rewire themselves.

II. Machine code / Assembly Language

- Machine code instructions still depend on the computer's architecture, but the variation isn't as great; many CPUs manufactured around the same time or by the same company will use the same machine code sets, in fact.
- Assembly language is a symbolic presentation of machine code so that people (very dedicated people with lots of free time) can read programs written in it.
- Most assemblers (programs that convert assembly code to machine code) support labelling and macros to make assembly language programming easier.

Some examples of assemblers are:
Looping control structures, simple data structures and some types!

III. Low-level Programming Language

- Formerly known as high-level programming languages.
- These languages have looping constructs, procedures, functions, some typing – the trappings of modern programming languages.
- Big improvement over assembly language.

Some examples of Low-level programming are:
FORTRAN, COBOL, BASIC, & arguably C:

IV. High-level Programming Language

- These are very convenient, but also very far removed from the computer they are running on.
- As a result, they typically aren't as efficient.
- They still may not be portable: implementation dependence. Java has had some problems with this.

Some examples of High-level programming are:

Java, Python, ML, Prolog, MATLAB, etc.

APA References

Penn, G. (n.d.). Levels of Programming Languages. Retrieved January 8, 2022, from https://www.cs.toronto.edu/~gpenn/csc324/lecture2.pdf#:~:text=Levels%20of%20Programming%20Language%20%E2%80%A2High-level%20Programming%20Language%20%E2%80%93e.g.%3A

Program Compilation | Learn Four Stage Program Compilation Process. (2021, August 21). Learn Computer Science. https://www.learncomputerscienceonline.com/program-compilation/#:~:text=Program%20Compilation%20Stages%20The%20conversion%20of%20program%20source