Universidad Politécnica de Yucatán

Structured Programming

Unit 1

Stages of program compilation and levels of programming

Mario Alejandro Canche Mex

Teacher Luis Gerardo Cámara Salinas

Robotics 2A

Stages of program compilation

There are six stages when we compile a program in any language and are these lexical analysis, syntax analysis, semantic analysis, intermediate code generation, code optimization and code generation.

First the compilation starts with a lexical analysis and this does two subprocesses of “scanning” and “screening”. Scanning involves finding substrings of characters that constitute units called textual elements. These are the words, punctuation, single- and multi-character operators, comments, sequences of spaces, and perhaps line boundary characters. In simple words is a scanner that finds the substrings and classifies each as to which sort of textual element it is. The next subprocess is screening and this involves discarding some textual elements, such as spaces and comments, and the recognition of reserved symbols, such as the key words and operators, used in the particular language being translated. It is the output of this process, usually called a token stream, that is the input to the parser.

The second phase is syntax analysis or parsing. The parser uses the tokens that the before phase there was created to create a tree-like intermediate representation that represent the grammatical structure of the token stream. This tree shows the order in which the operations in the assignment are to be executed. A typical tree that is used is the syntax tree in which each interior node represents an operation and the children of the node represent the arguments of the operation.

The third phase is semantic analysis, and this uses the syntax tree and the information in the symbol table to check the source program to check if there are semantic errors in the source program. Also, this collects the information about the types for the next phase of generation code, in this uses the hierarchical structure determined for the phase syntactic analysis for to identify the operators and operating the expressions and prepositions.

The fourth phase is intermediate code generation. In the process of translating a source program into target code, a compiler may construct one o more intermediate representations, the syntax trees are commonly used during the syntax and semantic analysis. After the syntax and the semantic analysis of the source program, many compilers generate a representation to the target, this are intermediate representations of low-level or machine-like, and we can think of as a program for an abstract machine. There are two important properties that the intermediate representation must have and are:

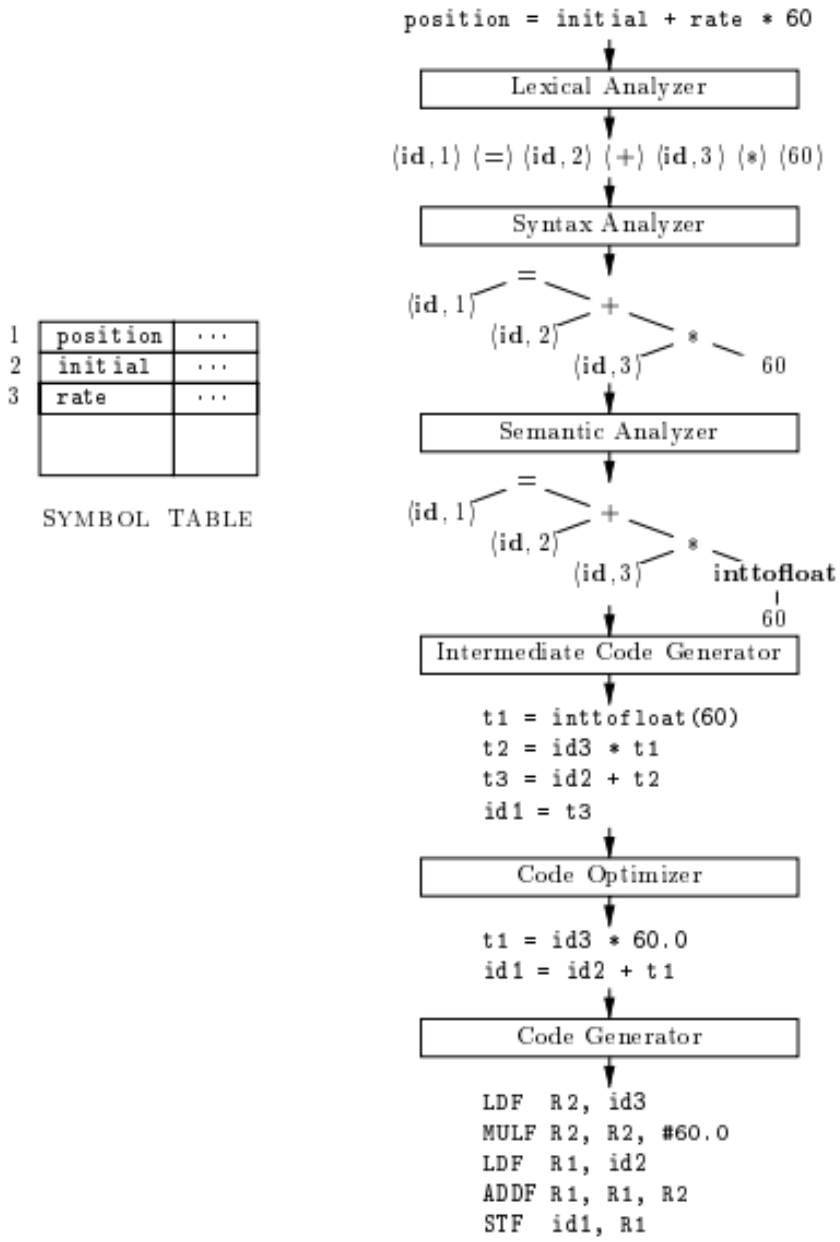
-Must be easy to produce.

-Should be easy to translate into the target machine.

The fifth phase is code optimization this tries to improvement intermediate code so that better target code will result. Usually better means faster, but other objectives may be desired, such as shorter code, or target code that consumes less power.

And the final phase is the code generation, this takes as input an intermediate representation of the source program and maps it into the target language. If the target language is machine code, registers or memory locations are selected for each of the variables used by the program. Then, the intermediate instructions are translated into sequences of machine instructions that perform the same task. A crucial aspect of code generation is the judicious assignment of registers to hold variables.

Representation of the stages of program compilation



Levels of programming.

Machine code

Is the essential language of a machine, also with this the machine interpretate code systems for the microprocessor. This is the unique language that a machine understands. The binary system is the unique language in the machine code.

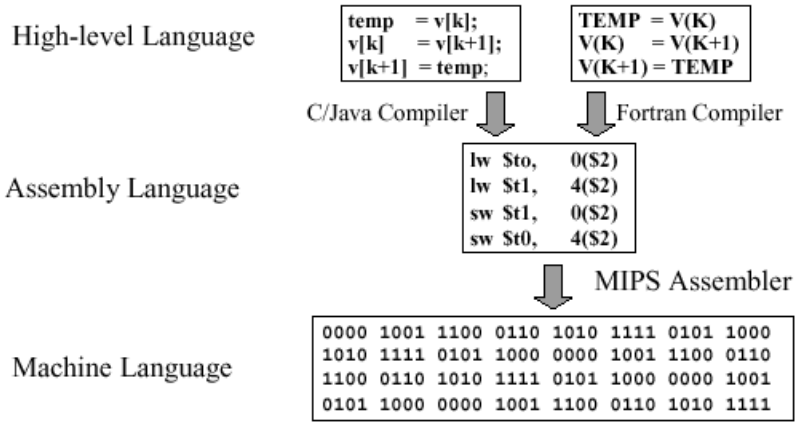
Low level language

Low level languages are used to write programs that relate to the specific architecture and hardware of a particular type of computer. They are closer to the native language of a computer (binary), making them harder for programmers to understand.

The example for this is assembly language, is a low programming language for a computer, or other programmable device, in which there is a very strong correspondence between the language and the architecture's machine code instructions.

High level language

High level languages are written in a form that is close to our human language, enabling to programmer to just focus on the problem being solved. No knowledge of the hardware is needed as high-level languages create programs that are portable and not tied to a particular computer or microchip. These programmer friendly languages are called 'high level' as they are far removed from the machine code instructions understood by the computer. In this we have more examples like C++, Python, C, JavaScript and others. In this case we need a compiler program to communicate with the machine.



References

Kantorovitz, I. P. (2004). *Lexical analysis tool*. ACM Sigplan Notices, 39(5), 66-74.

Malcolm Tatum. (2021). *What is Machine Code?*. May 5th, 2021, de EasyTechJunkie Sitio web: https://www.easytechjunkie.com/what-is-machine-code.htm

Aho, A.V., R. Sethi, and J.D. *Ullman, Compilers, Principles, Techniques, and Tools, Reading*, Massachusetts: Addison-Wesley, 1986.

Benjamin Archer. 2016. *Assembly Language For Students*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA.