Software Synthesis for Embedded Processors

A seminar paper for Special Emphasis A - Hardware/Software Codesign

Yashodhan Vishvesh Deshpande
Hamm-Lippstadt University of Applied Sciences, Lippstadt, Germany
yashodhan-vishvesh.deshpande@stud.hshl.de

***Abstract*—In this digital age we require computing large amounts of data for many smart applications. Complex computing is required in embedded devices like smartphones, computers and other electronic devices. This creates a demand for software synthesis on a large scale. This seminar paper includes various aspects related to software synthesis in embedded processors. This paper includes components like software synthesis in c programming and method in which compilers work.**

***Index Terms*—compilers**

I. Introduction

Software synthesis is necessary in many realizations such as automobile and electronic appliances. Many of these appliances are embedded systems with complex hardware. They have control systems, multi processor and memory chips. All these components can be controlled by software synthesis. This bring the physical system closer to the software systems.

A. High level programming languages

High level programming languages are languages that are in human readable form. These languages are designed for humans to easily program. Examples of these languages are Java and Phython.

To program in these we need Integrated development environment IDE. These environments are used to program and debug the source code. They can also be used for simulation purposes.

The processor cannot understand these languages directly. Hence compilers are used to translate these high level languages into low level languages. The processor only works in low level languages like machine language that is in binary numbers. The machine language is not a human readable language.

II. Compilers

Compilers are software applications that take source code as an input of a particular programming language. Generally this input source code is in high level programming language. This input source is then translated into low level programming language code. Mostly it is binary code that the processor requires to perform various operations. [?]

A. Processes in a compiler

1) Lexical analysis: This analysis is carried out by the compiler to identify tokens in the source code such as keywords and operators.

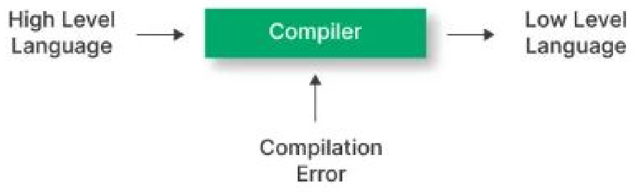


Fig. 1. Compiler operation [?]

2) Syntax analysis : The compiler checks every single line of code and the order in which the code is written. It checks to see if the code follows the official syntax rules of the programming language.

3) Semantic analysis: In this analysis the compiler checks the code for variable declarations. It checks to see if the variables are declared correctly.

4) Intermediate representation code generation: In this process the compiler transforms the code into another code with more concise code while maintaining all the functionality and same logic .

5) Optimization: In this step the compiler transforms the code into a that is more optimal in terms of its performance.

6) Output code: This is the last process. Here the compiler translates the optimal code to the binary machine language or any other intended low level language.

B. Compilation of C code

In order to obtain the accurate machine code the compiler takes care of a few critical process features like error checking, efficient execution and portability.

The code compilation is carried out in various steps for smooth operation of the compiler. The steps are as follows.

Pre-processing is the steps taken by the compiler before it starts to compile the code it includes removing comments, macro expansion and file inclusion.

1) Removing Comments: The first step is to remove all comments form the code. The comments in programming are used for the programmers explanation of the code only. They are used by typing //. They are not included in any logical operations in the code. So the compiler removes the comments whiles it compiles the code.

2) Macro Expansion: The compiler then does macro expansion to replace all the macros with there actual numerical value where ever they are used.

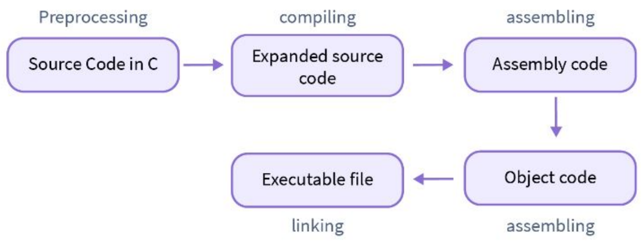
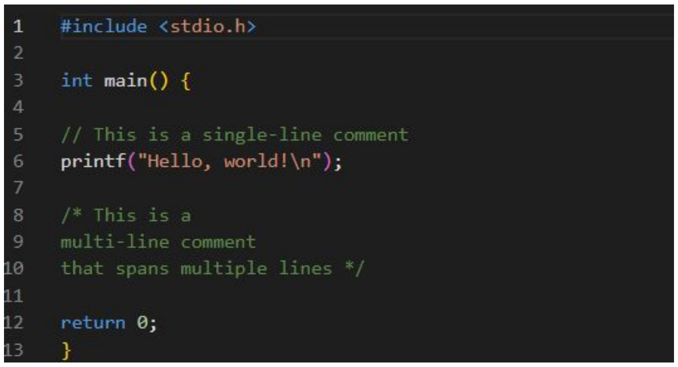


Fig. 2. Compiler Processing [3]



```
#include <stdio.h>

int main() {

// This is a single-line comment
printf("Hello, world!\n");

/* This is a
multi-line comment
that spans multiple lines */

return 0;
}
```

Fig. 3. Comments in C programming [?]

Macros in C are texts that indicate a numerical value and can be used in the code for easier coding by the programmer.

Example: #define PI 3.141592653589793

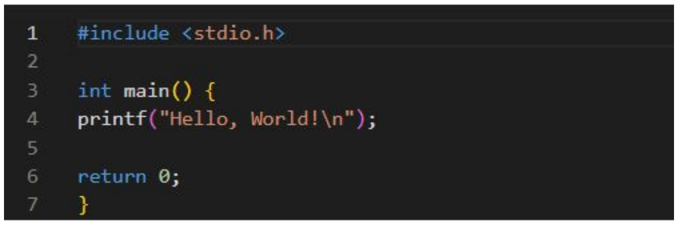
3) File Inclusion: In c programming external file and libraries are used for various applications to reuse certain sections of the code in the future. They are included in the source code by the programmer as header file(.h) backed up by the source file(.c). Example: #include ”math.h”

The compiler replaces the header files and library declarations with actual code within those files into the main source code. This gives our a single integrated source code without any extensions to outside files and libraries. [?]

4) Compilation: The compiler first converts the source code into the assembly language code. Then it changes the converted assembly code to a more efficient assembly code while maintaining full functionality of the code. At last the compiler generates object files that have .obj extensions. The object files are generated for the specific kernel it is intended to run on. During the whole compiling process the compiler also check for syntax errors.

5) Assembler: The .obj file created in the compilation process are sent to the assembler. The assembler converts the assembly code into the machine code.

6) Linking: The linker ensure that the final generated code is in the correct order. Sometime the final generated machine code is too long and need to be saved in two separate file. The linker links this two files using a linking module. It can also link some large header files to the final machine code. [1]



```
#include <stdio.h>

int main() {
printf("Hello, World!\n");

return 0;
}
```

Fig. 4. Simple C code for hello world [?]

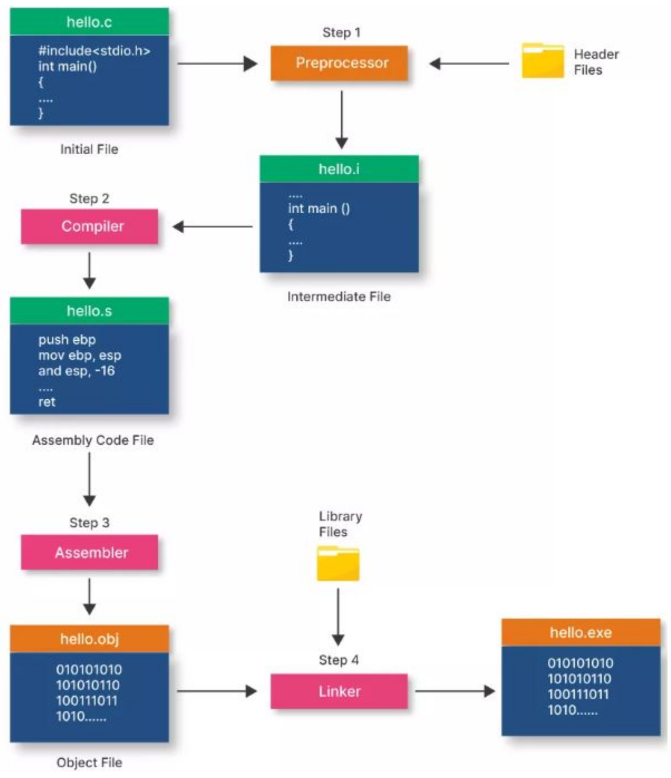


Fig. 5. Simple Hello Wold compiling Process [?]

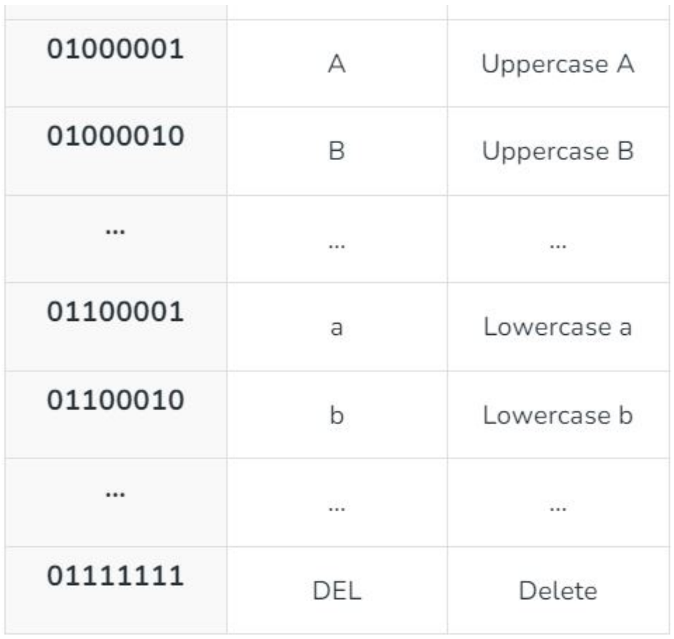
III. American Standard Code for Information Interchange

The ASCII is a standard used for coding a character in a high level language. It is a standard that defines decimal values in high level languages to the respective binary value. Hence it provide the basis for the high level programming languages.

Some features of ASCII are every single character has 7 to 8 bits binary value. The language is primary in English. [2]

IV. Conclusion

Software synthesis plays an important role in operating embedded systems. This is because they are a pathways for communication between programming logic designed by humans to be used in processor operations. This paper covers compiler operation and every step that the compiler takes before creating the machine code. Also it talks about in different tasks in converting C program to machine language. Some aspects of theASCII Standard are discussed.



01000001	A	Uppercase A
01000010	B	Uppercase B
...	...	...
01100001	a	Lowercase a
01100010	b	Lowercase b
...	...	...
01111111	DEL	Delete

Fig. 6. ASCII Standard Example [2]

REFERENCES

[1] Compilation in c: Detail explanation using diagram & example // unstop.

[2] What is ASCII - a complete guide to generating ASCII code. Section: Programming.

[3] Abhishek Chandra. Compilation process in c.