Programming Language

CSE-204-F

Lecture 1 & 2

Introduction

After studying this chapter, the student should be able to:

- ❑ **Describe the evolution of programming languages from machine language to high-level languages.**

- ❑ **Understand how a program in a high-level language is translated into machine language.**

9-1 EVOLUTION

To write a program for a computer, we must use a computer language. A computer language is a set of predefined words that are combined into a program according to predefined rules (***syntax***). Over the years, computer languages have evolved from ***machine language*** to ***high-level languages***.

Machine languages

In the earliest days of computers, the only programming languages available were **machine languages**. Each computer had its own machine language, which was made of streams of 0s and 1s. In Chapter 5 we showed that in a primitive hypothetical computer, we need to use eleven lines of code to read two integers, add them and print the result. These lines of code, when written in machine language, make eleven lines of binary code, each of 16 bits, as shown in Table 9.1.

The only language understood by a computer is machine language.

Table 9.1 Code in machine language to add two integers

Hexadecimal	*Code in machine language*			
$(1FEF)_{16}$	0001	1111	1110	1111
$(240F)_{16}$	0010	0100	0000	1111
$(1FEF)_{16}$	0001	1111	1110	1111
$(241F)_{16}$	0010	0100	0001	1111
$(1040)_{16}$	0001	0000	0100	0000
$(1141)_{16}$	0001	0001	0100	0001
$(3201)_{16}$	0011	0010	0000	0001
$(2422)_{16}$	0010	0100	0010	0010
$(1F42)_{16}$	0001	1111	0100	0010
$(2FFF)_{16}$	0010	1111	1111	1111
$(0000)_{16}$	0000	0000	0000	0000

Assembly languages

The next evolution in programming came with the idea of replacing binary code for instruction and addresses with symbols or mnemonics. Because they used symbols, these languages were first known as symbolic languages. The set of these mnemonic languages were later referred to as assembly languages. The assembly language for our hypothetical computer to replace the machine language in Table 9.2 is shown in Program 9.1.

i

The only language understood by a computer is machine language.

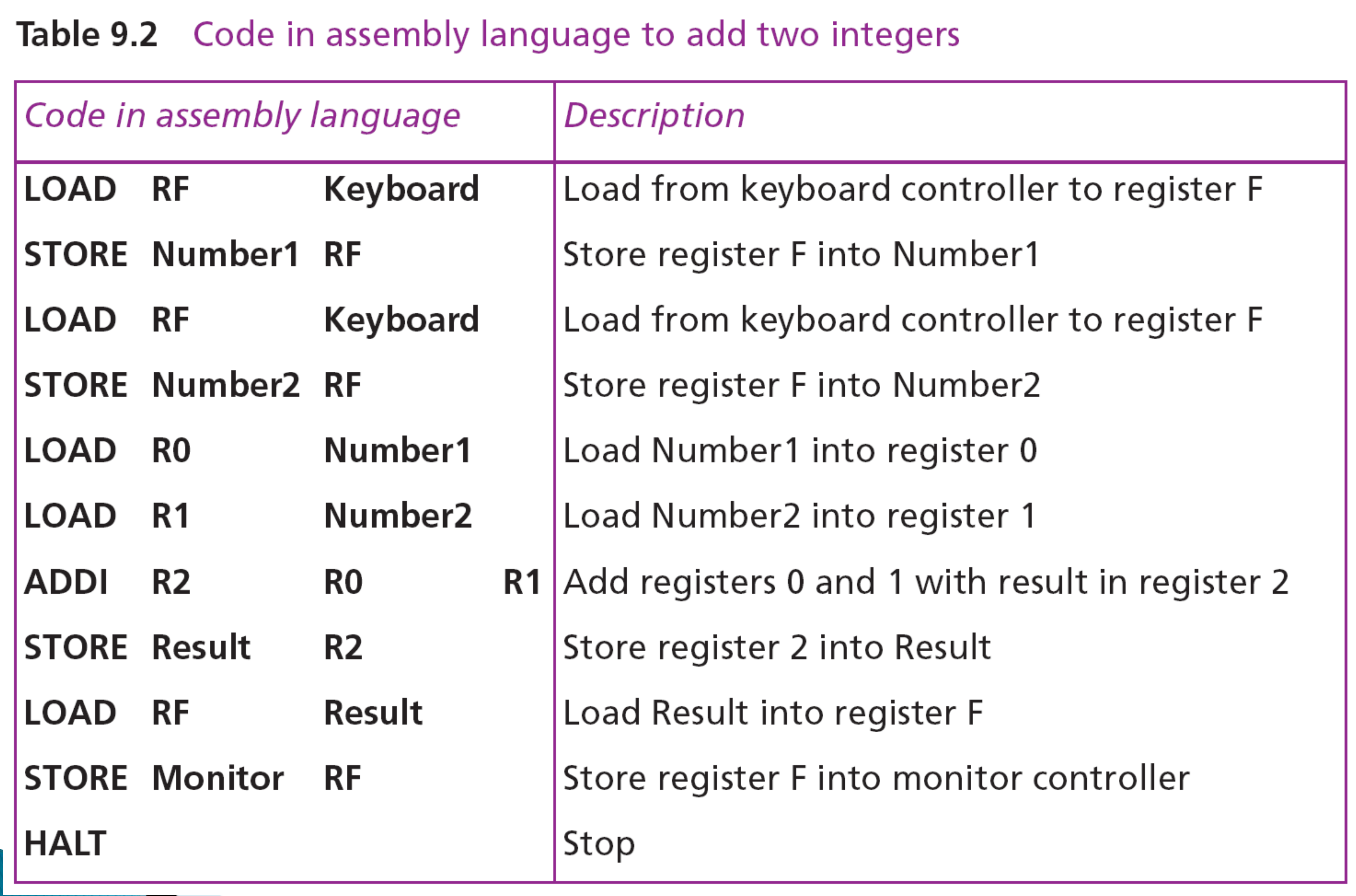

Table 9.2 Code in assembly language to add two integers

Code in assembly language				*Description*
LOAD	**RF**	**Keyboard**		Load from keyboard controller to register F
STORE	**Number1**	**RF**		Store register F into Number1
LOAD	**RF**	**Keyboard**		Load from keyboard controller to register F
STORE	**Number2**	**RF**		Store register F into Number2
LOAD	**R0**	**Number1**		Load Number1 into register 0
LOAD	**R1**	**Number2**		Load Number2 into register 1
ADDI	**R2**	**R0**	**R1**	Add registers 0 and 1 with result in register 2
STORE	**Result**	**R2**		Store register 2 into Result
LOAD	**RF**	**Result**		Load Result into register F
STORE	**Monitor**	**RF**		Store register F into monitor controller
HALT				Stop

High-level languages

Although assembly languages greatly improved programming efficiency, they still required programmers to concentrate on the hardware they were using. Working with symbolic languages was also very tedious, because each machine instruction had to be individually coded. The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high-level languages.

Over the years, various languages, most notably BASIC, COBOL, Pascal, Ada, C, C++ and Java, were developed. Program 9.1 shows the code for adding two integers as it would appear in the C++ language.

Program 9.1 Addition program in C++

```
/*      This program reads two integers from keyboard and prints their sum.
        Written by:
        Date:
*/
#include <iostream.h>
using namespace std;
int main (void)
{
        // Local Declarations
        int number1;
        int number2;
        int result;
        // Statements
        cin >> number1;
        cin >> number2;
        result = number1 + number2;
        cout << result;
        return 0;
} // main
```

9-2 TRANSLATION

Programs today are normally written in one of the high-level languages. To run the program on a computer, the program needs to be translated into the machine language of the computer on which it will run. The program in a high-level language is called the source program. The translated program in machine language is called the object program. Two methods are used for translation: **compilation** and **interpretation**.

Compilation

A compiler normally translates the whole **source program** into the **object program**.

Interpretation

Some computer languages use an interpreter to translate the source program into the object program. Interpretation refers to the process of translating each line of the source program into the corresponding line of the object program and executing the line. However, we need to be aware of two trends in interpretation: that used by some languages before Java and the interpretation used by Java.

Translation process

Compilation and interpretation differ in that the first translates the whole source code before executing it, while the second translates and executes the source code a line at a time. Both methods, however, follow the same translation process shown in Figure 9.1.

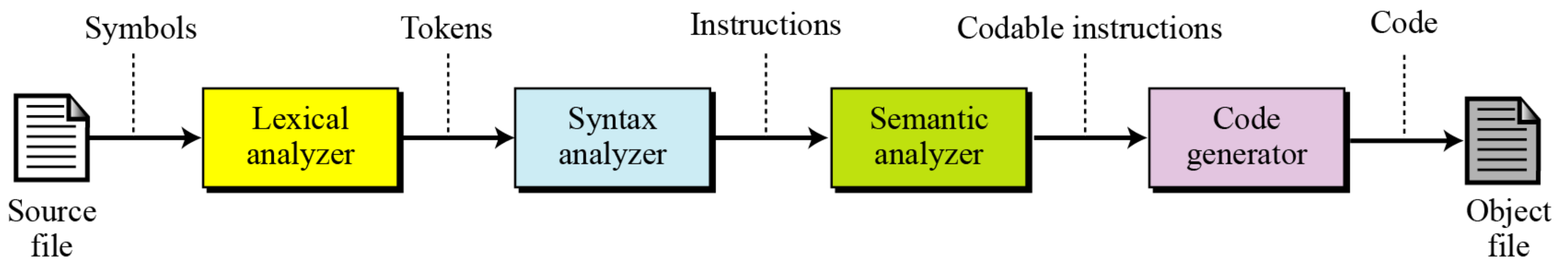

Figure 9.1 **Source code translation process**