Development of an Educational Computer System Simulator Equipped with a Compilation Browser

Yoshiyuki Miura
Faculty of Technology
Tokyo University of Agriculture and Technology, Japan
y-my@cc.tuat.ac.jp

Keiichi Kaneko
Faculty of Technology
Tokyo University of Agriculture and Technology, Japan
k1kaneko@cc.tuat.ac.jp

Masaki Nakagawa
Faculty of Technology
Tokyo University of Agriculture and Technology, Japan
nakagawa@cc.tuat.ac.jp

Abstract: We designed a simple but adequate educational high-level programming language, EL21, and developed an educational computer system simulator, ED21, by extending a traditional simulator with the addition of a compiler, an editor and a compilation browser. The compilation browser is for visualization of the process and showing how programs written in a high-level programming language are translated into an assembly language. We conducted an experiment to evaluate the effectiveness of our system in an introductive course of computer science in our university and verified its effectiveness in enabling students to understand the execution principles of programming languages.

Keywords: Simulation-based Instruction, Compiler Visualization, Programming Education

1. Introduction

Recently, educational computer simulators have been widely used to teach operational aspects of computer architectures (Barua, 2001; Nishida et al, 2003; Johnson et al, 2002). However, these systems are too much inclined to focus on operational principles of hardware. Thus, it has been reported that students cannot understand how programs written in a high-level programming language are executed in actual computers even if they learn operational principles of computer hardware and high-level programming languages simultaneously (Decker et al, 2001; Evangelidis et al, 2001). To address this problem, Decker et al. developed a system that visualizes how a piece of a program is translated into an assembly-language code (Decker et al, 2001). However, the system is only able to visualize the compilation process of assignment statements. Hence, it is insufficient for students to understand the whole process of program translation followed by execution in a computer. Therefore, in this study, we have developed an educational computer system simulator that can completely visualize the process of translation of programs written in a high-level programming language. The simulator, named ED21, is designed to support the introductive course of computer science for freshmen in our university and several other universities.
2. ED21

2.1 System Configuration

ED21 is based on ED9900 (Yamaguchi et al., 2000) and it is developed by adding the following extensions:

- Design of an educational high-level programming language, EL21
- Implementation of a tiny editor, EE21
- Implementation of a compiler, EC21

Because ED9900 was implemented by using JavaApplet, we have also adopted it for these extensions.

In our system, a user first edits an EL21 program using EE21. Then EC21 translates it into an assembly code for ED21. Finally, the user can observe the execution process of the program using ED21. Figure 1 shows the overview of ED21. Extensions are described in detail in the following sections.

![Figure 1: Overview of ED21](image)

2.2 EL21

We designed a simple programming language, EL21, which is suitable for novice students to learn the execution process of programs written in it. Logo and Basic are famous programming languages for educational purpose (Jones, 2003; Dann et al., 2000; Krumholtz, 2003). However, Logo is developed for young children and it is not adequate for the programming activities of undergraduate students. Basic requires a run-time library which makes program execution opaque. Therefore, we decided to design a new language. The specification of EL21 was decided by referring several procedural-programming languages. It was also taken into consideration that students will have to study some conventional programming languages.

Because ED21 is designed for novice students, we aimed to make the language specification of EL21 so simple that it can be understood at a glance. However, we also aimed that EL21 be sufficiently enriched in language constructs so that users can enjoy programming to some degree. For these reasons, EL21 has five types of statements; that is, it permits “assignment”, “if”, “while”, “read” and “print” statements. In general, conventional programming languages contain the “for” statement. However, the syntax and semantics of the “for” statement are much more complicated than those of the “while” statement. Additionally, any flow control for repetitions in a program can be implemented in our framework. Hence we have excluded the “for” statement from the EL21 specification. We have
also excluded the function call because of complicated semantics concerning arguments passing. Figure 2 shows the language specification of EL21.

<program> ::= <program name> <compound statement>
<program name> ::= <identifier>
<statement> ::= <compound statement> | <if statement> | <while statement> | <read statement> | <print statement> | <assignment statement>
<assignment statement> ::= <variable name> = <expression>; 
<while statement> ::= while ( <expression> ) <compound statement>
<read statement> ::= read(<variable name>); 
<print statement> ::= print(<expression>);
<compound statement> ::= { <statement sequence> }
<statement sequence> ::= <statement> <statement sequence>
<variable name> ::= <identifier>

Figure 2: Language specification of EL21

2.3 EC21

EC21 is a compiler for translation of a program written in EL21 into an assembly-language code for ED21. EC21 consists of four phases: lexical analysis; syntax analysis; intermediate-code generation; and object-code generation. The lexical analyzer takes a simple approach based on backtracking. The syntax analyzer adopts the predictive recursive descendant parsing algorithm. As the intermediate code, the parse trees are used. The object code generator uses a simple recursive algorithm. Though most conventional compilers have a code optimization phase, EC21 is designed not to have any code optimization phase. Because the naive code is much easier to understand for novice students and it is possible to ask them to perform hand-optimizing exercises of the compiled codes, we have selected this option.

2.4 EE21

ED21 users may not be familiar with file operations. So we have prepared an editor EE21 which runs inside the ED21 system. That is, users who have not learned much about file concepts can edit a program and translate it into a code by using EE21. The lower left window in Figure 1 shows EE21.

Users can edit EL21 programs by using EE21, and after completion of programming, they transfer the programs to EC21. EE21 can also edit the assembly code for ED21. Therefore, EE21 has the function to read and write the code and the data between its text field and the memory area of ED21. Because of this function, users can edit assembly-language codes and optimize them by themselves.

3. Compilation Browser

3.1 Objective

As described in the previous chapter, any program written in EL21 is translated into the assembly code for ED21 through EC21. The compilation browser visualizes this translation process in EC21. It aims at enabling users to understand the following points:

- A compiler translates a source program into an assembly code which will be executed by the computer.
• The translation process includes lexical analysis, syntax analysis and code generation phases.
• The translation can be executed systematically.

Note that we do not intend to illustrate the strict translation process by the compilation browser. If a user recognizes the existence of a compiler and its role, we take it as a success.

3.2 Visualization of Compilation Process

In this section, we explain the contents visualized by the compilation browser. The compilation browser visualizes the processes of EC21. EC21 consists of four phases. These are lexical analysis, syntax analysis, intermediate-code generation, and object-code generation. Among these phases, we visualized lexical analysis, syntax analysis and object-code generation phases, but omitted the intermediate-code generation phase for simplicity. After completion of programming, pushing the “EC21” button in EE21 makes the program transferred to EC21. Then the compilation browser is invoked and a new window is opened. Translation process for this program will be displayed in the window. Figure 3 shows the execution process of the compilation browser.

As in Figure 3, the execution results of the three phases are displayed in this order in the compilation browser by clicking the corresponding buttons. The code generation phase in the compilation browser generates the object codes that can be executed on ED21.

4. Evaluation

To evaluate the effectiveness of ED21, we conducted an experiment for 75 freshmen in the computer science course of our department at the end of the first semester in 2004. Before the experiment was conducted, they had just finished the course of “Introduction to Programming” where C programming was taught with exercises. The students were divided into two groups. Group A learned about the computer system by using ED21 while Group B learned about the computer architecture by using the conventional simulator. Before and after the experiment both the groups answered twenty-four questions about execution of programs. By comparing these results, we evaluated ED21. Table 1 shows the average scores.

<table>
<thead>
<tr>
<th>Group</th>
<th>before</th>
<th>after</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A</td>
<td>13.84(57.67%)</td>
<td>16.03(66.79%)</td>
</tr>
<tr>
<td>Group B</td>
<td>14.44(60.17%)</td>
<td>13.88(57.83%)</td>
</tr>
</tbody>
</table>

Table 1: Average scores of both the groups. The values in parentheses represent the ratio of correct answers.
As shown in Table 1, Group A shows notable progress that is judged significant by t-testing while Group B does not. From this table, we can conclude that our system ED21 is effective in enabling novice students to understand the execution principles of high-level programming languages.

5. Conclusion and Future Works

We have developed an educational computer system simulator ED21 by which users can learn not only hardware of a computer but also the execution principles of programming languages. For this purpose, we first designed the simple programming language EL21. Then we implemented ED21 by extending a traditional simulator with the editor EE21, the compiler EC21 and a compilation browser. As the result of an evaluation experiment, we have shown that ED21 is effective in enabling novice students to understand the execution principles of programming languages. Future work includes the following:

- Graphical visualization of the compilation browser instead of current text-based visualization; and
- Extension of our simulator so that users can recognize the existence of the operating system and its role.

References


