A Comprehensive Experiment Scheme for Computer Science and Technology

Lisheng Wang¹, Renzhi Huang²

¹, ²Department of Computer Science and Technology, Tongji University, Shanghai, China

Abstract - Nowadays, there are some problems in experiment courses for the students major in CS (Computer Science and Technology), such as experiment teaching is focused on validating experiments and the experimental contents are scattered. To solve these problems, this article puts forward a comprehensive experiment scheme. It guides students to design hardware components first, then achieve their own processor, build a compiler for an advanced language and design a simple operating system. Contents of the comprehensive experiments which are closely related with each other cover most of foundational and core courses in CS. The feasibility of the experiment scheme is still being studied currently, and part of the task has been completed. But some experiments are still lack of detail plan, on which further study is needed.

Keywords: Computer Science and Technology, Comprehensive experiment scheme, Experiment teaching

1 Introduction

CS (Computer Science and Technology) is a highly technical and practical specialty. While attention is paid to the theory teaching, the experimental teaching also can't be ignored. The scientific and systematic experiments for teaching not only strengthen the students' theoretical foundation, but also help them transfer professional knowledge into the ability for practical application, improve the ability of solving engineering problems and then raise innovative spirit. Colleges and universities are paying more and more attention to the experiments in teaching arrangement. Though the experiment teaching has been strengthen, actually sometimes the effect is not very satisfactory. There are still some problems, such as, with lesser difficulties, the validating experiments are in the majority. When conducting an experiment, the students don't need to understand the principle behind, just finish the experiments according to the experiment guidance. Besides, experimental contents are scattered. Experiments between different courses are unrelated. They are based on a single knowledge, and can't bridge different knowledge to help students find out the importance of certain knowledge points and set up a complete concept of computer system.

In view of the experiment teaching situation and problems described above, this paper proposes a comprehensive experiment scheme for the students major in CS, to help them consolidate the basis theories, also get familiar with the concrete application of certain theory and make innovation in the experiments [¹].

2 Basic idea of the comprehensive experiment scheme

This comprehensive experiment scheme aims to design hardware components first, implement a processor, then write an advanced language compiler and a simple operating system. In undergraduate education, experiments in this comprehensive experiment scheme are divided into each course, including Digital Logic, Principles of Computer Organization, Microcomputer Principle and Interface Technology, Assembler Language Programming, Compiler Construction Principle and Operating System. This comprehensive experiment scheme follows two principles. The first is that experiments in different courses correlate with each other. Results of the experiments in pre-requisite course will be used for the experiments in successor curriculum. For example, basic logic components such as register which is designed in experiments of Digital Logic will be used in experiments of Principles of Computer Organization to construct a processor, besides the compiler designed in experiments of Compiler Construction Principle will compile the operating system designed in experiments of Operating System. The second principle is that the comprehensive experiment scheme is focused on designing experiments, and advocates innovation. As compared to validating experiments, designing experiments are more difficult. But if given enough guidance, the students would spend more time to complete the experiments, so the experiments will get better effect. Furthermore, innovation is advocated in experiment conducting. The students are allowed to add new ideas to the experiments [²].

3 Comprehensive experiment design

3.1 Digital logic component design

Been widely used in every field of science, Digital Logic is a practical and important course. As a series of basic experiments in the comprehensive experiment scheme, digital logic component design includes combinational logic and temporal logic component design. Encoder, decoder, data selector, adder and numerical comparator are the typical combinational logic components to be designed, while flip-flop, shift register, synchronous and asynchronous counter are
the temporal logic components. There are two ways to implement the digital logic components described above. First, after getting familiar with one hardware design tool, such as ISE design suite supplied by Xilinx, the students draw the circuit of the component as input to finish the design. The second way is that, the students learn a hardware description language, and then use the language to describe the component to finish the design.

In these hardware foundational experiments, there are three tasks to be finished. First, the students should be skilled in the use of one hardware design tool. Second, a hardware description language should be mastered. Then, the logic components stated above would be designed and validated.

3.2 Assembler language programming

The assembly language is a kind of programming language related to the specific system architecture. In view of MIPS32 instructions are open and neat, and the MIPS32 assembler can be got from the Internet easily, the MIPS32 architecture will be introduced in the specific teaching arrangement. Students learn the instructions' classification and format, addressing ways about MIPS32 architecture, and the design method about assembler language programming in class. Experiments for this course are based on MIPS32 simulator. The experimental contents are writing different assembler language programs to satisfy different requirements. For example, designing programs to calculate complex mathematical expressions can help students be familiar with the arithmetic and logical operation instructions, while designing programs to compare two strings can help students be familiar with the conditional jump instructions and recycling program design method. Besides, designing a sort or search program using subroutines can help the students master the thought of designing program in modular.

In these software foundational experiments, there are two tasks need to be completed. First, after getting familiar with the basic instructions and register organization of MIPS32 architecture, the students should know how to use the MIPS32 simulator. Then, students should be skilled in writing assembler language programs and validate them on the MIPS simulator. Experiments for this course lay a foundation for the later processor design. The students who are interested can write BIOS (Basic Input and Output System) for the later designed computer.

3.3 Computer components design and processor integration

There are also series of experiments for Principles of Computer Organization in this comprehensive experiment scheme. Along with theory teaching, guiding the students to implement kinds of computer components could help the students deeply understand the composition structure and work principle of computer components. And it will be helpful in processor designing. The experiments of computer component design are based on MIPS32™ architecture that the students master in the experiments of Assembler Language Programming. Using the logic components designed before, the students can describe the computer components in a hardware description language. The components to be designed include ALU, barrel shifter, multiplier, divider and controller. At last, all kinds of components will be integrated into a full processor, and which would be validated [5].

Tasks to be finished in these experiments are listed below. First, finish all kinds of computer components design. Then learn the method of processor design and integration with computer components and basic digital logical devices [4]. Flexible experiment arrangement can be taken to adjust the difficulty. For example, the students can be divided into groups, so that each group finishes the experiments cooperatively.

3.4 Periphery connection parts design and system integration

The standard computer system includes ALU (Arithmetic Logic Unit), controller, memory, and input/output device. Processor only consists of ALU and controller. In order to facilitate the interaction between users and the processor, also to set up a complete computer system of hardware level, the bus module, interrupt control module and some commonly used I/O modules need to be added. To adjust the difficulty, the bus arbitration function will not be considered, leaving the processor as the only master equipment. Besides, less number of interrupt sources and simple interrupt priority is considered in the design of interrupt control system. Input and output module design can take GPIO (General Purpose Input/Output) and UART (Universal Asynchronous Receiver/Transmitter) as examples. After the completion of the above design, they will be integrated with the processor into a complete computer system. Using some EDA (Electronic Design Automation) tools, the system can be implemented on a specific FPGA development board [5].

As to perfect the computer hardware system, experiments in this section need to finish the following tasks. First, complete the bus module design. Second, design the interrupt control system. Third, write the GPIO and UART module. Last, integrate the computer hardware system, and implement it on a specific FPGA development board.

3.5 Compiler building

Compiler Construction Principle is a quite theoretical course. If there is no experiment, the students cannot find out the practical utility of this course. It must become boring. To help students build a more profound understanding about compiling principle, experiments in this section aim at designing a programming language compiler to translate advanced language programs into binary programs based on the completed hardware architecture. The big experiment task can be divided into many small stages, including lexical analysis, syntax analysis, symbol table management, intermediate code generation, object code generation and
code optimization \(^6\). Lexical analysis stage finishes an advanced language lexical scanner, while syntax analysis stage finishes a syntax parser according to the output of the lexical scanner. As a data structure to save the information of all symbols defined in the source program, the symbol table needs to be well designed. Intermediate code generator finished in the intermediate code generation stage, generates particular intermediate language code. Then the particular intermediate code will be translated into binary code in object code generation stage according to the instruction format and register organization of target machine. The code optimization stage is responsible to the goal of code optimization \(^7\). Considering the difficulty of building a compiler, it is suggested that the students can be divided into groups to complete the task cooperatively.

3.6 The operating system experiment design

As a core course in CS, Operating System covers many great human ideas. If there is some way to make these ideas concrete, it would be of great help to students in understanding the essence of the operating system. Experiments in Operating System use the computer hardware system designed in pre-requisite courses’ experiments as a platform. The students will use MIPS32 assembly language and an advanced language that can be compiled by students' compiler, together with principles of operating system learned in class to design a simple operating system or some function modules. Specific experimental contents are still in exploration. But the general design principle is that while controlling the difficulty in experiment; try best to help students design a simple but complete operating system.

4 Summary

A comprehensive experiment scheme is proposed in this paper. Following it, the students design hardware components first, and implement a processor. After the hardware is built, an advanced language compiler and a simple operating system will be designed. This comprehensive experiment scheme is mainly focused on designing experiments. It guides the students to finish series of experiments related with different courses, so as to strengthen the students’ theoretical foundation, and to help them improve the ability of solving engineering problems, furthermore to cultivate their design concept and spirit of innovation. Now, exploration about the comprehensive experiment scheme is ongoing. A processor based on MIPS I instruction set has been built. Some periphery components and the compiler are being designed. But there are still some drawbacks. A detail plan cannot be proposed for operating system experiment design. More efforts are needed.

5 Reference


