Explorations and Practice on Teaching Methods of Introduction to Computer Systems

Haoyu Zhang[1,a,*], Hu Jin[1,b], Jiafang Wang[1,c]

[1]College of Computer Science and Big Data, Heilongjiang University, Harbin, Heilongjiang Province, China

[a]15364542@qq.com, [b]1993006@hlju.edu.cn, [c]1993010@hlju.edu.cn

*Corresponding author

Keywords: Introduction to Computer Systems, IA-32 Assembly Language, Systems Competency, Learning Assistant Software

Abstract: The IA-32 assembly language is a crucial part of Introduction to Computer Systems. Mastery of the IA-32 assembly language directly impacts students' systems competency. The paper has successively attempted teaching research methods such as compiling a learning document, recording instructional videos, and designing a learning assistant software in order to enhance teaching effectiveness, improve students' learning efficiency, stimulate their learning motivation, and sharpen their practical skills. The students' performance of the IA-32 assembly language has improved, through the above methods, especially after using the learning assistant software. The next step will involve further refining and promoting this learning assistant software to achieve even better results.

1. Introduction

Following the arrival of the post-PC era, the model of cultivating talents in the computer science domain has slowly changed from concentrating on students' programming development skills to highlighting their systems thinking and abilities[1,2]. Computing Systems Competency means the capacity to deliberately utilize a systems viewpoint to comprehend the wholeness, interconnection, hierarchy, dynamism, and openness of a computer system, and to use systematic methods to understand the mechanisms of how hardware and software cooperate and interact[3]. In response to this broad disciplinary shift,the Software Engineering program in the College of Computer Science and Big Data at Heilongjiang University established a new compulsory course,Introduction to Computer Systems. Centered on the developmental and executional trajectory of high-level language programs,the course is intended to cultivate students' systems capability. Into a single instructional sequence it brings several key dimensions of computer systems—data representation and operations,machine-level program representation,and program execution—so that students can gradually form,in cognitive terms,a relatively complete hierarchical framework of computer system architecture. Once the overall structure of computer systems is understood and foundational system principles are grasped,students' capacities in software design,software development,and system application can be strengthened further[4,5].

For the purpose of attaining the teaching goals mentioned above, the instructors of the course have chosen "Introduction to Computer Systems (2nd Edition)" written by Professor Yuan Chunfeng from Nanjing University as the teaching material. Through actively learning from Professor Yuan's teaching group via the "Chinese University MOOC" platform[6], and on the basis of the teaching plan of our college and the particular learning situations of the students, we have worked out the syllabus for the "Introduction to Computer Systems", designed the content of theoretical and experimental teaching, as well as the assessment methods, and put together a series of teaching documents like course lesson plans and lab manuals. By applying these teaching materials in our educational practice, the initial outcomes have been obtained[7].

Nevertheless, within the teaching summary, the course instructors did still pinpoint certain issues that emerged throughout the teaching process. The most frequently encountered problem is that

  DOI: 10.25236/ehmit.2025.034

students possess an insufficiently profound understanding of the IA-32 assembly language. As a result, this brought about a rather shallow comprehension of the subsequent topics: the machine-level representation of C programs, the allocation and access of complex data types, along with out-of-bounds access and buffer overflow. This insufficiency was clearly demonstrated by remarkably low scores on questions related to the exam[8]. In an attempt to enable students to rapidly master the IA-32 assembly language, and in this way, get a hold of the subsequent knowledge points concerning the relationship between high-level languages and IA-32 assembly language, and to improve students' system capabilities at the earliest opportunity, the course instructors have taken up and put into practice teaching reform approaches such as putting together learning materials, filming instructional videos, and creating learning assistant software.

2. Compile Learning Materials for IA-32 Assembly Language

Because the Software Engineering program at our college does not provide the "Assembly Language Programming" course, students have no previous experience with IA-32 assembly language before taking the "Introduction to Computer Systems". Nevertheless, "Introduction to Computer Systems (2nd Edition)" only touches briefly on the register organization, addressing modes of IA-32 assembly language, and the use of basic instructions in the instruction set. Consequently, students usually find it hard to know how to begin when learning IA-32 assembly language.

To tackle this problem, the course teachers jointly put together the "Assembly Language Learning Guide". The guide first centers on the functions of 80X86 registers, the form of AT&T assembly language, addressing mechanisms in IA-32 assembly, the IA-32 assembly instruction set, and assembly language programming. For example, in the part about addressing modes, the guide has a sample program that shows all the addressing modes one by one. Along with memory snapshots showing where data is stored, it helps them see all at once how we get back our data using every different way.

Moreover, in terms of dealing with the IA-32 assembly instruction set, it includes some usages of those instructions, which are written down using RTL so that we could have a better understanding about how they work. In the same way as when we teach on the part where there’s a lesson about IA-32 assembly language programming and add a total program example to solve this problem of a very few examples given by the book and they’re not complete, it is just codes. Because we have ensured that all programs in the guide can run on Ubuntu 16.04, so all our code example is correct, student can learn IA-32 Assembly Language by themselves.

In the second part of the guide are programs which exemplify 3 major aspects: the machine level view of a C program, Allocation & Access of complicated Data Types, and Out Of Bound access with buffer overflow. They are shown here compared side-by-side with the original C program and its IA-32 assembly version. So as to make it evident for anyone reading that nearly all the statements in the C code should have a matching entry on the assembly list. And there’s also some program where I’ve added diagram which shows the stack allocation while executing. This kind of visual helps make the students understand how a stack works better and it’s easier for them to understand such concept. Take explaining how it works with passing the argument by reference during procedure call as an example, we construct a particular kind of stack frame which assigns special address places for different parts. It is very easy to see that the addresses of the parameters can be written into the stack when we teach, and students will understand it immediately.

Classroom teaching of IA-32 Assembly Language is based on the study guide. Through the example problems within it, students will be able to learn about the way of IA-32 assembly programming much quicker. Also, there were lectures on how programs can be presented at the machine level. The relevant program examples taken from the guide, also it is proved to work.

3. Record Instructional Videos for IA-32 Assembly Language

Developing students' systems-related ability is not just about making them have a theoretical

understanding of the connections among the internal abstract levels of a computer system. It also demands that they experience the links between these levels in real-world practice. For this purpose, apart from putting together the "Assembly Language Learning Guide", the teachers have also created a large number of teaching videos. These educational videos mainly include guides on assembly language itself and comparative research between assembly and C language, and there is also a relatively small quantity of videos about relevant hardware ideas.

3.1. Assembly Language Instructional Videos

Instructional videos about Assembly language mostly focus on 3 big things: addressing methods, use of basic assembly instructions and examples of how to develop a simple program in an assembly language. Adhering to the concept of emphasizing practical use, teachers steered clear of the usual way of teaching which was through classrooms with slides. Instead of this, they made those videos just by programming in plain old assembly code on Linux. From the editing, compilation, linkage to debugging, executing etc., every single step of making up the assembly code is shown. So as to enable the student to exactly retrace the whole programming process by watching the teaching video.

By taking the clarification of the unconditional jump instruction JMP as an example, the most important thing is to make clear the way by which the JMP instruction redirects the execution sequence to a specified place, which helps the target instruction to be carried out smoothly. From the point of view of instructions, the main concept of this mechanism is that the address part of the JMP instruction contains the offset, that is, the difference between the address of the target instruction and the address of the instruction immediately following the JMP.

In the video tutorial which offers an explanation of jump instructions, initially, we created an assembly program with the JMP instruction in a Linux environment. And then we used the as and ld tools to put it all together so we could run it. At last we debug with the help of GDB in order to see the sequence of execution to the student. It can be clearly seen that the JMP instruction is diverting control to some address in order to run its target instructions.

Then, we used the objdump tool to show the disassembled code. This makes it possible to clearly see the number that is being stored in the address part of the JMP instruction. Combining it with the known addresses of the target instruction and the one that follows the JMP, it’s easy to figure out that the difference between them is exactly the same as what we see in the address part of the JMP instruction.

3.2. Assembly Language and C Language Comparative Instructional Videos

In creating our comparisons on how to assemble, and teach a class in C, we also did an example where we took some sample C sourcecode, and converted it into assembly. In this kind of procedure the students can find out which parts of codes in both languages have equal function and make it much easier to compare. In the end, it is equivalent to saying that we teach people "how to fish" instead of just giving them a fish so they are able to find solutions on their own.

Like with how we’re looking into the machine-level showing of the c if -else statement, we have a definite method that we use. First of all, I make a really simple C programme that includes an if -else. After that we use GCC to convert this into a program in assembly language. At last， we have made a corresponding relation for each C statement to its counterpart in the assembled program.

The condition is because inside C there’s an if else statement and it might contain some code which would be ran whenever the conditional test in the "if" portion turns out to be true. Therefore the conditionals within the c source code do not have a one-to-one match up with what you would find as conditional jumps within your generated assembly instead they tend to be inverted. It makes it much more likely that mistakes will be made by students carrying out code-completion tasks and so becomes a big problem for people trying to learn.

When the C code is read as "if (a > b)", the corresponding assembly code will be "jle L2". This circumstance should be explicitly and emphatically explained to students.

Videos recorded by the instructors of the course encompass: basic knowledge of the IA-32

assembly language, programming methods in the IA-32 assembly language, instances of the machine-level representation for procedure calls in C programs, instances of the machine-level representation for selection statements in C programs, instances of the machine-level representation for loop statements in C programs, instances of array allocation and access, instances of struct allocation and access, instances of union allocation and access, as well as buffer overflow attacks. Besides making videos about topics associated with assembly language knowledge points and the comparative analysis between assembly language and C, the course instructors have also made use of simulation software to generate videos covering topics related to computer hardware. Heilongjiang University has its own self-contained learning platform. The recorded teaching videos are made available to teaching classes via this platform for students to view and learn from. These videos act as a significant addition to classroom instruction, allowing students to thoroughly and at any time learn the key ideas of the "Introduction to Computer Systems".

4. Development and Implementation of Learning Assistant Software

As the teaching process advanced, the course teachers noticed that even though the learning guide and teaching videos had boosted students' effectiveness in theoretical learning, they hadn't notably enhanced students' practical skills. No mutually strengthening relationship was set up between theoretical study and practical use. After an analysis, the course teachers came to the conclusion that a main cause of the less-than-ideal learning results was that students weren't skilled enough in operating the command-line interface of the IA-32 assembly language in the Linux environment. This lack of efficiency notably decreased their enthusiasm for learning.

So, the course teachers made a decision to create a learning support software that gets rid of the need for command-line operations in the Linux environment. The software can act as an easy-to-use integrated development environment for IA-32 assembly language. It smoothly combines different programming phases, such as editing, compiling, linking, debugging, and execution, into an integrated work process. At the same time, it includes important knowledge points of IA-32 assembly language in the software, enabling students to easily look up concepts they've forgotten during the programming. This design guarantees simplicity and high-efficiency, thus increasing students' learning output and promoting a mutually strengthening relationship between theoretical comprehension and practical application.

4.1. Development Tools for Learning Assistant Software

The learning assistant software's 1.0 version is going to be developed for the Windows environment. Once it reaches a mature state, it will then be ported to the Linux environment. According to this plan, the course instructors have made a decision to utilize Qt as the development tool for the learning assistant software. Qt is a cross-platform C++ application development framework which is designed for graphical user interfaces. It supports a large number of compiler systems, it can also be installed and run under the windows system or the linux system. It has an internal cross platform feature which makes sure that the same code base gets compiled and executed across various system with out any changes, it will also adjust easily on a different platform.

The learning aid program is developed with the development version being Qt version 6.4.2 The qt sdk has everything you will ever need to develop cross-platform applications. In all its tool kits, MinGW can be said to be very important. This is this particular tool kit, it can be used to make apps with Linux-like look on Windows.

After you have installed the qt on your windows system, you will have a simulated linux enviroment with out install any thing else. In this kind of environment the learning assistant software will be able to run on Windows. This can also decrease the dependence of student on the command-line in linux, as well as the operational errors which would occur when they are learning. Therefore, students can improve their learning efficiency.

4.2. Functional Design of Learning Assistant Software

A piece of software, which is intended to be used as an aid in learning, is going to be called HIA for short. It is both a comprehensive environment that can be used to develop the IA-32 assembly language, and it is also a very good reference material for understanding the knowledge of the IA-32 assembly language. Therefore, It has all basic function of an integrated Development environment:

The very first place, the very top of the primary interface, we find a menu bar divided into 4 parts; they are named File and Edit and Run and Help. The File has 5 items under it: new, open, save, save as, exit. Subordinate item is mainly responsible for the creation and preservation of assembly languages, and also the shutdown process of software. In the "Edit" segment there are three sub-items called cut, copy and paste. The subordinate things allow one to do such actions as cut, copy, and paste with respect to the assembly language file which happens to be currently open for editing. In the "Run" segment, there are three subordinates: generate executable, run program and debug program. The subordinates deal mostly with the process of transforming assembly language program into its corresponding object files and executable file, also running the executable file and debugging it. The Help segment is about the introduction and clarification on how to use this learning support software.

In the middle part of the software window, there are four windows set up in two rows. The upper one is further divided into the left， middle and right parts. The left part is the navigation area which gives a list of sample programs and explanation about instruction regarding to IA-32 assemly language. The middle portion is the program editing window where we will be making changes in our source code of the Assembly Language. Right Section: Instruction – display window to show the explanation on the instruction. In the lower part of the middle section is what we call the Console window, which displays all the messages produced throughout the entire course of compilation and also shows all kinds of information regarding errors that occur when changing our assembly source code into executable files.

Taking into account that there might be some students who do not know about the structure of an IA-32 assembly language program very well, we can just write the codes following the example programs in the navigation window. In navigation window, there are examples programs given as well as explanation on instructions which is relevant to IA-32 assembly language

In order to do this indexing, I have taken some examples in our guide to the Assembly Language and made a separate file with explanation for each instruction. And these file will be joined as resource to the learning assistant softward. Use the tree-widget list feature of Qt to show the information about examples programs and the instructions index on the navigation window.

And then if the students click with the mouse on one of the examples shown in the navigation window, the source code for that example will appear in the Program Editing Window. Then let the student change the code on this and have them write their own program, after that they should use the Save As option within the File menu to save the source code of it. In addition it should be known that the file extension of IA-32 assembly languge programs has to be .s or else some following operation might have problems.

After we have done with our editing work, we will move on to compile. Assembly language: If we talk about the assembly language then its compilation process is also done in 2 stages. In the first place, the assembler(as) is employed to generate object files out of a source text composed in assembly language. In the second step we use the linker(ld) to generate executable file from our object files The above both two stage carried on with selecting the "Generate executable" in Run.

The learning assistant software retrieves the name of the file when it saves the file of the assembly language program. Also this filename is shown up on top of menu bar for the student so that he can be sure about what program is he going to compile. By depending upon this file name, software makes the creation of object files and executable files with same name. And meanwhile, we can see what information is made by the compiler while compiling our code in the console window.

In the case that there are syntax mistakes in the assembly language program, the learning

assistant software would not have the capability to produce the proper object file. However，it can also show these error message on the console widow. According to what the console tells us, we can tell that there is an error in some line or statement of our source code. Then we can use the instruction index in the navigation bar to read the usage guide of the defective instruction in the instruction display area on the right hand side and correct it immediately.

If we suppose that our object file is created without errors now it's time to make an executable file. At this moment in time, the Console Window only displays some output info produced along the way as we create our Executable File. As for the fact that we're working on IA - 32 Assembly Language programs, it means that when it comes down to actually running these programs, we make use mainly of what's called a Debug Program. When the programmer selects "Debug program" option then a new pop up window like windows command prompt appears by the learning assistant software. Within that very window GDB is started and debugging on the executable file takes place. And this environment can support all the GDB command operations, which will help the programmer see what changes have been made to registers and memory stacks, etc., by looking at the results after running the program.

4.3. Shortcomings of Learning Assistant Software

Learning assistant software is the teaching and research result of the 'Introduction to Computer systems' course group. It integrates the teaching material well with the development environment, giving the students a good place to both learn about things and practice them. It can lower the threshold of learning and help students start learning IA-32assembly language quickly. Its purpose is to encourage students' interest in learning through the idea of doing rather than being afraid of assembly language programming. As a result, it can improve the level of students' system capability and meet different levels of students' learning needs.

But because the developers do not have much experience in software creation, it is inevitable that there would be some flaws with how this software was made as well as its functions. Like the learning assistant software does not have shortcut icons on its interface which makes it hard to use for students. Also, there is no display of line numbers on the source code in the editing window. It's hard to find where the error is on a single line when there are mistakes with this kind of programming. Then at the next stage will be the course teachers trying to better those two.

5. Conclusion

In the teaching and learning process for the 2023 group of Software Engineering majors, the learning guide, instructional videos, and the learning assistant software have been put into use. When comparing the final exam scores between the 2022 and 2023 groups of Software Engineering majors, it was found that the average score for course goals related to the IA-32 assembly language went up from 37.712 in the 2022 group to 38.391 in the 2023 group. The achievement level of course goals also got better, rising from 0.712 in 2022 to 0.724 in 2023, which shows some positive teaching results.

The course teachers will keep collecting feedback from students. Based on this feedback, they will make the software's features more refined and better, and then use these improvements in the teaching and learning of the 2024 group of Software Engineering majors. Moreover, the course teachers intend to combine the instructional videos with the learning assistant software to make its functions further enhanced. The aim is to keep improving and reach a state that is almost perfect, and finally, to make the software more widely applied and promoted.

At the same time, the course teachers plan to further polish the teaching materials. They will use these materials as a basis to create a new textbook that can better match the teaching and learning activities of our teachers and students.

Acknowledgements

This paper represents the phased outcomes of two projects. One is the 2022 Heilongjiang

Province Higher Education Teaching Reform Research General Project titled "Research and Practice on the Visualization of the 'Introduction to Computer Systems' " (Project No.: SJGY20220191), and the other is the 2022 Heilongjiang University Higher Education Teaching Reform Project also named "Research and Practice on the Visualization of the 'Introduction to Computer Systems' " (Project No.: 2022B12).

References

[1] Chun-feng YUAN, Shuai WANG.(2013)University Computer Science Education Should Emphasize the Cultivation of the "Systems Perspective".China University Teaching, 12, 41-46.

[2] Zong-li JIANG.(2011)Professional Competency Composition and Cultivation for Computer Science Majors.China University Teaching, 10, 41-46.

[3] Wei-min ZHENG.(2021)System Competency Development for Computer Science Majors.China University Teaching, 5, 19-23.

[4] Ling LU.(2022)Discussion on Developing Problem-Solving Skills through Introduction to Computer Systems.University Education, 7, 136-138,148.

[5] Shi YAN.(2024)Retrospect and Prospect on Computer System Competency Cultivation.IT Education, 4, 1-6.

[6] Chun-feng YUAN, Zi-hao YU, Guang-hui ZHU.(2023)Teaching Philosophy and Course Resource Development for Introduction to Computer Systems.IT Education, 11, 12-17.

[7] Hao-yu ZHANG, Cheng-guo LV, Yang GAO.(2023)Explorations and Practice on Course Construction of Introduction to Computer Systems.Heilongjiang Education(Research and Evaluation ofHigher Education), 6, 75-77.

[8] Jian HE, Shou-bao SU, Xiao-hui MO.(2017)Teaching Exploration of Assembly Language Courses Based on a Systems Perspective for Computer Science Majors.IT Education, 9, 98-101.