A NOVEL APPROACH FOR TEACHING SYSTEM SOFTWARE BY INTEGRATING WITH OTHER COURSES

Nagarathna D Kulenavar[1], Shameeda Begum T[2]

[1]*Associate Professor, Department of Computer Science and Engineering, BVBCET, Karnataka, India*
[2]*Assistant Professor, Department of Computer Science and Engineering, BVBCET, Karnataka, India*

Abstract

Teaching individual courses in our curriculum is done with inadequate focus on integrating the learning with other courses. This results in students not appreciating the courses showing less interest in learning process. This paper presents the new approach developed by us to teach the system software theory and practical courses for the computer science students. In this approach we are integrating the System Software course with the learning in the course on Microprocessor 8086, Microcontrollers and its Applications. Further this paper elaborates as how the use of a hypothetical machine in conventional teaching gave the freedom to teach system software concepts but failed to connect to reality. The method followed in this new teaching approach is that it relates all the concepts taught in system software with the real-time system/machines instead of hypothetical machine. It provides the flow that correlates the course with the existing software. This paper also explains how the changes in technology helped to consider real-time machine as an example while teaching concepts of system software rather than hypothetical machine. This paper also through light on the limitations of teaching the System Software course with hypothetical machine, the challenges faced during the change in syllabus and a brief explanation on the chosen real-time machine. We have observed better learning of the concepts of systems software through this approach. The result analysis prove, connecting the learning in courses taught in early semesters in the delivery is observed to help in building the learning context. It has resulted in improving their performance and understanding in the course.

Keywords: *System Software, teaching approach, real-time machine, hypothetical machine, practical approach.*

----------***----------

1. INTRODUCTION

One of the most frequently encountered phenomena in education systems is that of curriculum change. Though it is enthusiastically welcomes by progressives and regretted by traditionalists it is accepted by teachers and administrators as one of the recurring features of educational life. If education is to fulfill it's most basic function that is technology change it has to respond to the changing pattern of syllabus [1].

The System Programming or System Software is usually a core course for computer science curriculum at under graduate level. The main purpose of the System Software course is to strengthen system programming skills of students. As there is a close relationship between system software and machine architecture undergraduate students will be taught the design and implementation of different system software such as assembler, linker and loader [4].

Limitations of existing System Software course is that it was taught using hypothetical machine as a result students cannot correlate it with real time existing machine. Also studies have found that market appeal/industry demand is one of the most important factors affecting machine choice in computer science education [3]. Therefore change in teaching methodology is required.

In this paper we proposed a new teaching methodology with revised course content by replacing hypothetical machine to real time machine to study System Software. The new syllabus (course content) now includes 16-bit 8086 microprocessor. The new syllabus overcomes the limitations of old syllabus by helping students to develop system software for real time existing machines. This approach found to have a creative impact on students.

The System Software course and its corresponding lab is introduced at the sixth semester level which is of four credits with two minor exams, semester end exam and course project. The questions of minor and semester end exam are framed to test the learning levels of students according to the blooms taxonomy. The course along with the delivery and evaluation method addresses technical outcome of the program.

2. IMPORTANCE OF SYSTEM SOFTWARE COURSE IN COMPUTER SCIENCE CURRICULUM

The computer science engineer must have skills in networking, database management, system programming, assembly language programming, and web designing and so on. The system programming skill is incorporated among students by teaching them courses such as System Software, Compiler Design and Finite Automata and Formal languages (FAFL).

Earlier System Software course was introduced in fifth semester which covers different hypothetical machine architectures, Assemblers, Loaders etc. As a fact any high level language program written in any of the programming language like C, C++ and Java is to be converted into assembly language program by compiler, the assembly language program is translated into machine level language by assembler and this machine level language is loaded in the memory by loader. Then any task execution is possible by bringing that task into memory. Thus assembling and loading are the two system processes which computer science engineers must know. In order to understand assembling, loading and other system related functionalities System Software course is introduced in Computer Science Curriculum.

3. CONVENTIONAL TEACHING

The system software like assembler, linker and loader are all dependent on machine architecture, therefore we need to first select a machine and teach it's architecture in order to make students understand assembling, linking and loading process.

In early days system was a black box and people had very less knowledge about the system. So, System Software was taught conventionally using model machines or systems called hypothetical machines such as Simplified Instructional Computer(SIC) and Simplified Instructional Computer/extended(SIC/XE). These hypothetical machines are not real time machines they just simulate and help us understand the assembling, linking or loading process. They include very less complexity which in turn makes system software designing, implementation easy and even fast.

As depicted in the Fig-1, the input for the SIC or SIC/XE hypothetical machine is the assembly language program of that machine. This assembly language instruction set is simple and is used only to show the simulation and not the actual working in real world. This assembly language program is translated into the machine level language based on the architectural features and instruction set of the hypothetical machine. Thereby the converted machine language program is linked and loaded in to the memory for execution.

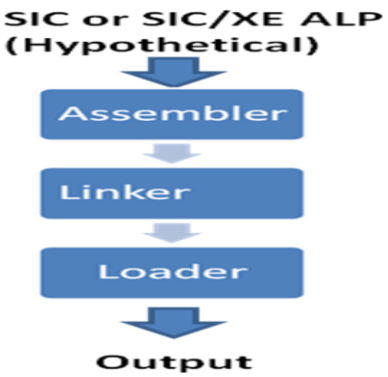

Fig -1: Conventional Teaching using hypothetical SIC or SIC/XE machine

4. LIMITATIONS OF CONVENTIONAL TEACHING

In reality there are many real time machines which already exist and students of our institution would have studied some of the real time machines architecture in their earlier semester courses like Computer Organization and CSDA. This induces us to think, why to teach System Software course using hypothetical machine? It is not possible to realize real time problems and it failed to connect to reality. Conventional teaching has following other limitations:

- Unable to analyze exiting system software code of any real time machine.
- Complexity in understanding other machine architectures.
- Complexity in implementing algorithms of real time machine system software.
- Learning hypothetical machine is of little use.
- We are unable to practice assembly language programs on already existing assemblers like MASM, TASM, and NASM.

5. PARADIGM SHIFT

It is thus realized that change in teaching approach is essential. Therefore hypothetical machines are replaced by real time machine as shown in Fig-2

To overcome the limitations of conventional teaching, System software course is to be taught by means of real time machine. Thus there was a transformation from hypothetical to real-time machine to study system software. As depicted in the Fig-2, the input for the real time machine is the assembly language program of that machine. This assembly language program is translated into the machine level language based on the architectural features and instruction set of the chosen real time machine. Thereby the converted machine language program is linked and loaded in to the memory for execution.

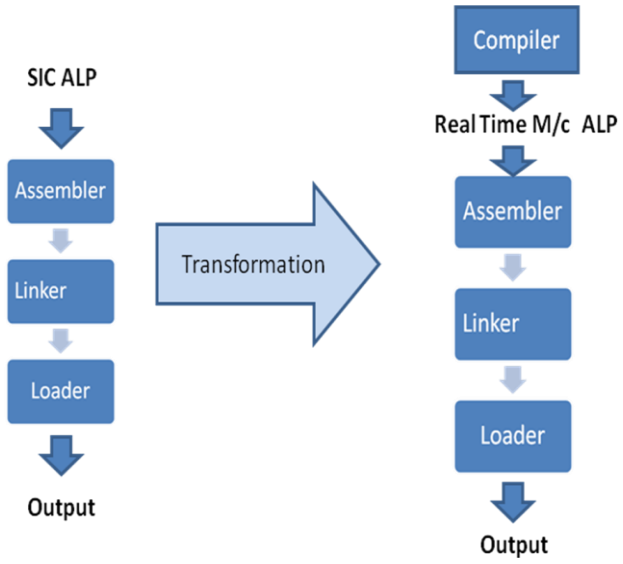

Fig -2: Transformation from hypothetical to Real-time machine

6. NOVEL SYLLABUS

Now, our current syllabus includes 8086 Microprocessor as a real time machine to demonstrate, design and develop system software, which is a novel approach to teach system programming.

8086 is a general purpose Microprocessor used in many applications. 8086 is chosen as it provides a varied instruction set and has several advanced architectural features designed to support multiprogramming and multiprocessing [2]. Segmentation is the concept which is introduced in 8086 machine which in turn form base for an Object Oriented concepts and even for the subjects where segmentation is used.

6.1 How it Worked?

The replacement of hypothetical with 8086 machine worked very successfully till end of the semester. It teaches students regarding designing of real-time machine's system software and their architectural features. In addition, the course builds skills in programming, code reading/analysis debugging and independent learning skills.

6.2 Challenges in New Syllabus

1. Difficulty and time consuming in understanding existing machine architectural features:

   Example1: Memory in SIC is 2 to the power 15 bytes. Knowing only size is all about memory details of SIC. But 8086 microprocessor memory is totally different which has segmented memory. Hence it is difficult and time consuming.

   Example2: Most of the instructions of SIC or SIC/XE uses either one or two instruction formats where as 8086 microprocessor's single instruction uses one to six instruction formats which increases implementation complexity[5].

2. Difficulty in the implementation of pass1 of two pass assembler algorithm for an existing machines:

   Example1: Most of the instructions of SIC have fixed size. So, pass1 algorithm in SIC is implemented by adding size to generate addresses. And in SIC/XE, there are some markers to identify the size of the instructions. Depending on the marker its corresponding size is added to generate addresses in pass1 implementation. Where as in 8086 Microprocessor or in any other real time existing machines, instructions size is not fixed and even there are no markers to identify the size of the instruction [5]. So, it is difficult to implement 8086 Pass1 assemblers.

6.3 Evidences

As a part of course projects each group of students were assigned other machines like Microcontroller and 8085 microprocessor to carryout case study. Students were able to designed and implemented assemblers for these two machines successfully. Teaching system software with 8086 real time machine helped them to bring out similarities and differences with other real time machines and understand system software of those machines. Result analysis also shows significant growth in their performance, as shown in Chart-1.

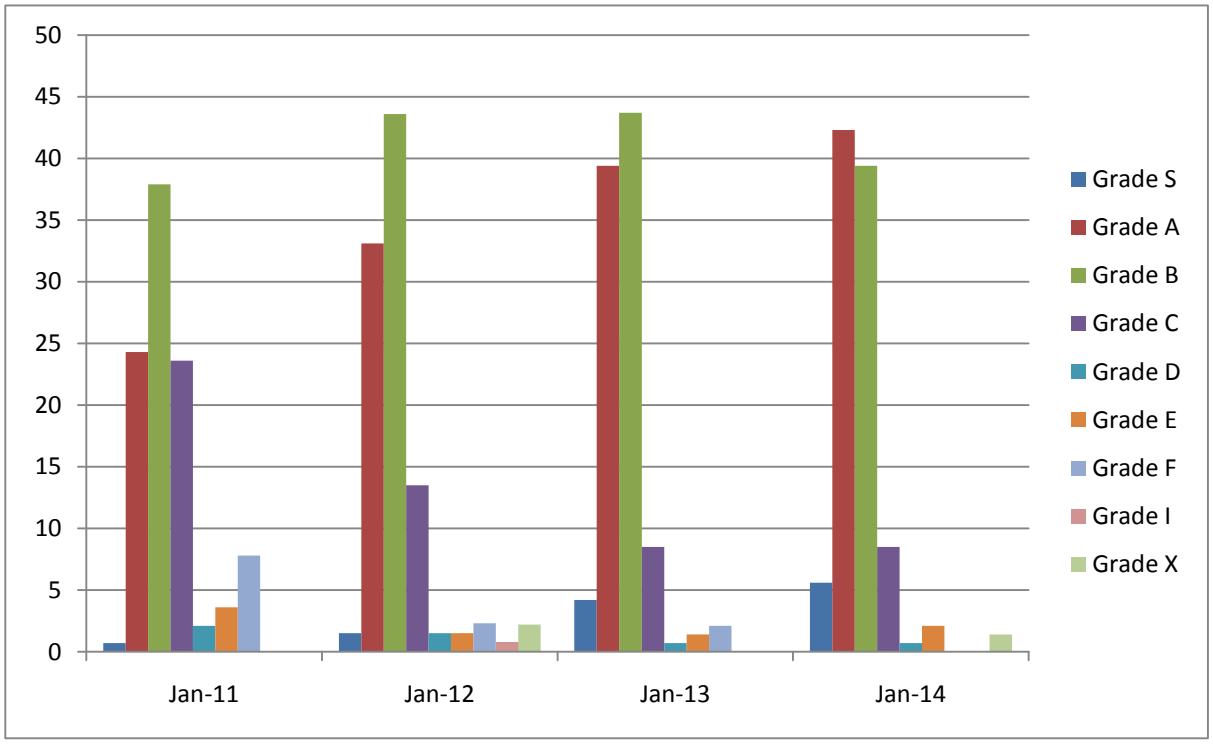

Chart -1: Result Analysis, X axis - Batch of student, Y axis - Number of students in percentage

Table -1: Grade equivalent percentage

S Grade	A Grade	B Grade	C Grade	D Grade	E Grade	F Grade
=>90%	75-89%	60-74%	50-59%	45-49%	40-44%	<40%

7. CONCLUSION AND FUTURE WORK

This novel approach to **t**each System software for existing real time machines gives more practical knowledge, improves thinking ability, makes other existing software code analysis and adaptation from one machine to other machine easy.

The overall outcome indicates that the adoption of real time machine has significantly been encouraging in terms of students overall growth and develops a kind of thinking they are likely to find valuable. Majority of the students have expressed their satisfaction over the changes introduced in syllabus.

As a future work 8086 Microprocessor can be replaced by either 8051 Microcontroller which in turn have its own application in embedded systems or any other more advanced processors.

ACKNOWLEDGEMENTS

The authors wish to thank Department of Computer Science and Engineering of B.V.Bhoomaraddi College of Engineering and Technology. This work was supported by Prof. K.R. Biradar(Head of the Department, Computer Science and Engineering), Principal Dr. Ashok S. Shettar(BVBCET, Hubli, Karnataka, India) and Prof. GopalKrishna Joshi. Our sincere thanks to all those who are directly or indirectly involved in this work.

REFERENCES

[1] Editorial Introduction: The process of curriculum change by S. JOHN EGGESTON, Keele.

[2] MICROPROCESSOR SYSTEMS: THE 8086/8088 FAMILY: Architecture, Programming, and Design, 2nd Ed. By Yu-Cheng Liu and Glenn A. Gibson.

[3] M. de Raadt, R. Watson, and M. Toleman. Introductory programming: what's happening today and will there be any students to teach tomorrow? In Proceedings of the 6th Conference on Australasian Computing Education, pages 277–282. Australian Computer Society, Inc., 2004.

[4] Leland. L. Beck and D. Manjula: System Software, 3rd Ed, Pearson Education, 2007.

[5] Douglas V. Hall: Microprocessor and Interfacing, 3rd Ed, Tata McGraw Hill, 2007.

BIOGRAPHIES

Name: Nagarathna D Kulenavar
Designation: Associate Professor.
College: B.V. Bhoomaraddi College of Engineering & Technology
Area of Interest: Translators (Assemblers).
Contact No: 9844915127
Email-id: kulenavar@bvb.edu

Name: Shameeda Begum T
Designation: Assistant Professor.
College: B.V. Bhoomaraddi College of Engineering & Technology
Area of Interest: Translators (Assemblers).
Contact No: 9916076848
Email-id: shameeda@bvb.edu