2018

Lab Manual Design with Engineering Learning Style and Flipped Learning Model in Computer Engineering Technology Education

Yu Wang
CUNY New York City College of Technology

Sunghoon Jang
CUNY New York City College of Technology

Lab Manual Design with Engineering Learning Style and Flipped Learning Model in Computer Engineering Technology Education

Yu Wang and Sunghoon Jang

***Abstract*—We have designed a lab manual based on Felder-Silverman learning style model (FSLSM) and the flipped classroom model for engineering education. This lab manual is developed for the early junior year course of "Microcomputer Systems Technology" and emphasizes student-centered active learning experiences with practical exercises and open-ended questions. Instead of taking traditional assembly language to study computer architecture, we are looking for a different approach to teach students to learn the assembly language by embedding an inline assembly language module into a C program. Our lab guide consists of online videos and practical exercises using various platforms including Microsoft Windows OS, Linux OS, Microsoft Visual Studio, and Visual Studio Community. With this new approach, students will be able to design creative lab projects instead of following a lab procedure. Students are able to work on the platform using multiple programming languages (C/C++ and Assembly), and multiple hardware devices (PC or Laptop, x86 device, Linux). With this new lab manual design, we guide students to preparatory contents and materials before coming to class by various activities described in online videos and practice exercises, etc. This lab-learning approach combined with the principle of flipped classroom and engineering learning styles can provide additional opportunities to advance the students' engagement in the studies of computer engineering technology.**

***Index Terms*— C/C++, engineering learning style, flipped learning, inline assembly, lab manual design, microprocessor**

I. Introduction

It is well known that each student has a particular learning style and that each style should be accommodated by tailored instruction. The instructional methods that prove most effective for students with one learning style is not the most effective method for students with a different learning style [1]. The FSLSM (Felder-Silverman learning style model) [2][3] classifies engineering students as having preferences for one category or the other in each of the following four dimensions: (a) sensing or intuitive, (b) visual or verbal, (c) active or reflective, and (d) sequential or global. Having a framework for identifying the different types of learners can help an instructor formulate a teaching approach that addresses the needs of all students. The FSLSM theory proposes the hypothesis that engineering instructors who adapt their teaching style to include both roles of each of the given dimensions should be close to providing an optimal learning environment for most students in any given class. The recent pedagogy approach of the flipped classroom model reverses the traditional learning experience. This model is more focused on interactive group learning activities inside the classroom, and direct computer-based individual instruction outside the classroom [4][5][6][7][8]. The flipped model aims to provide more student active learning experience for professors to guide projects in the classroom by exposing students to preparatory content and material before coming to class. The preparatory materials usually include various media formats, such as online videos and practice exercises. The research works of [9][10][11] have shown that flipped classroom model and proper design instructions to teach engineering technology students more opportunities for engagement and achievement. The research [9] studied the course leadership formalized in the Microprocessors course via flipped classroom. Students who served leadership can develop better self-efficacy. They are able to identify their own strengths and recognize the strengths of their lab partner and other students in the class. The research work of Gehringer and Peddycord [10] shows that students in the inverted-lecture class of Computer Architecture and Multiprocessing exhibited high levels of engagement. The research by [11] shows that student satisfaction with the "flipped" model is related to student learning styles. However, the improved learning was not supported by empirical evidence for a C++ and Java object-oriented design and programming course. The advantages of a "flipped" approach are not found in the conceptual and factual knowledge gained by students. What are our effective approaches to design, coach, and deliver instructions, especially for lab class activities of Microcomputer Systems Technology course?

The emphasis on hands-on laboratories in regular computer engineering technology curriculum has varied over the years. Traditional lab manuals are designed to enhance and reinforce basic knowledge which students learned from theory. From the hands-on experience, engineering technology students gain first-hand evidence verifying what they have learned from the lecture class. Most of current laboratory manuals used by our freshmen, sophomore or early junior years focus on expository, problem solving, and discovery styles. They are cookie book styles to provide the procedures to guide students conducting the exercises and experiments. Students are mainly responsible to present the collected data in the forms of tables or graphs and correlate them to a particular theory, hypothesis or model they have learned from the regular lecture classes. Most of the professors at our Computer Engineering Technology department in the CityTech followed the traditional approach, where they spend the entire lab period with students to address procedural issues and repeat to explain the same mistakes to the students on different lab workbenches. With twenty-two students in a two and half hour lab class, the average time each student can receive from professor's individual instruction is approximately seven minutes. With such short time to deliver the instructions to each group of two students, efficiency is sacrificed.

We have a diverse student body. The students who enter our computer engineering technology program have different academic

This work was partially supported by PSC-CUNY Award #60310-0048 and National Science Foundation Noyce Award #1340007.

Dr. Yu Wang is with the Department of Computer Engineering Technology, the New York City College of Technology of the City University of New York, Brooklyn, NY, 11201 (email: YWang@citytech.cuny.edu)

Dr. Sunghoon Jang is with the Department of Computer Engineering Technology, the New York City College of Technology of the City University of New York, Brooklyn, NY, 11201, USA (email: SJang@citytech.cuny.edu)

backgrounds. For example, students' math levels range from fundamentals of mathematics (MAT 1175), college algebra and trigonometry (MAT 1275), to precalculus (MAT 1375), calculus I (MAT 1475), and calculus II (MAT 1575). It is a challenge for our faculty to design a lab class and apply effective teaching strategies. When faculties design and teach a lecture and a lab, it is necessary to incorporate engineering learning styles to accommodate diverse student education background to address all student needs, while giving individual student insights into their strengths and weaknesses.

In Section II, an assessment blueprint is introduced. In Section III, lab manual design with learning styles is discussed. In Section IV, examples of the exercises in lab manual are illustrated. Finally, in Section V, the conclusions of our work are summarized.

II. Design an assessment blueprint

Currently our program has an open-access mission and follows 2+2 model, associate degree in Electromechanical Engineering Technology (AAS EMT) + bachelor technology degree in Computer Engineering Technology (BTech CET). During the first two years, we followed the EMT curriculum, and during the second two years, we followed the CET curriculum. The two programs of AAS in EMT and BTech in CET have been accredited by ETAC/ABET[12]. Our department has an enrollment of more than one thousand students. Most of students will continue their BTech degree study after they complete their AAS degree in EMT. In each semester, our BTech CET program receives dozens of transfer students from other majors, such as the majors of mechanical engineering technology and electrical engineering technology. Many transfer students come from local community colleges such as Queensborough Community College, LaGuardia Community College, and many others. The enrollment illustrates that an increasing number of students are seeking our BTech CET degree.

In our curriculum, the Microcomputer Systems Technology course (CET 3510) is the first course for the students who study the upper level BTech degree. Studying the assembly language and its relationship to the microcomputer architecture is the core part of CET 3510. Our department chose CET 3510 as the critical course to be assessed in each year to satisfy the ABET criteria and Middle State process. In 2012, we updated CET 3510 assessment blueprint to ensure we reach every milestone of the course to build consistency to assess student learning outcome. The assessment blueprint (*TABLE 1*) has been used to directly measure the course objective and the student learning outcome via departmental exam since 2012. The assessment blueprint was later revised in 2014 and 2017.

We have assessed student performance in carrying knowledge, logical thinking, programming skills, mastery of concepts, and further towards an ability of creative application. A part of the assessment result showed that student learning outcomes (SLOs) to master the E, F, G, and H of *TABLE 1* were less than 75% overall in the years of 2012 and 2013, which is lower than the department target rate 75% as shown in *TABLE 2*. In the year of 2014, we changed our text books to Assembly Language for x86 Processors [13] and Intel Microprocessors [14] from High Level Assembly Language [15]. However, the exercises from these two textbooks [13][14] focus on either assembly language for the older DOS environment or Visual C++ express with assembly language for the Windows environment. Our students in the CET 3510 class had difficulty learning the Intel assembly programming and the Windows programming because most of them were just co-taking the C++ programming course. How can we design and develop our own lab manual to teach students to program a microprocessor and understand its architecture? How do we achieve a learning goal to match our students' background, learning styles, strengths, and weaknesses? We did all this by starting to develop a few lab handouts in 2014 and posted them in CityTech openlab website for lab classroom usage. We took a different approach to teach students to learn the assembly language by embedding an inline assembly language module into a C program. As a result, SLOs of E, G, and H were better than previous years, close to 75% or higher overall in the years of 2014 and 2015. The tailored instruction to match engineering student learning styles improved SLOs.

TABLE 1
CET 3510 ASSESSMENT BLUEPRINT

Student learning outcomes (SLOs)	
A	Knowledge of the components used in a computer system
B	Knowledge of data formats of signed numbers and unsigned numbers
C	Mastery of concepts on the selected principles of the computer architecture, especially as used in the Intel x86 family of microprocessors
D	Explain and analyze selected the principles of memory address, addressing mode, data structure and organization
E	Perform computer arithmetic operations in the machine level by integer arithmetic instructions and floating point arithmetic instructions
F	Write and utilize an assembly or a C/C++ language to gain insights into instructions
G	Design a bit mask and perform bitwise operations in an assembly or an C/C++ language
H	Develop an application programming (assembly or C programing) to interface and access computer hardware input and output ports (I/O ports)

TABLE 2
CET 3510 PARTIAL ASSESSMENT RESULT (2012-2015)

SLO	Spring 2012	Fall 2012	Spring 2013	Fall 2013	Spring 2014	Fall 2015
	Target rate 75%					
E	60%	74%	59%	70%	83%	82%
F	47 %	70%	71%	53%	65%	74%
G	74%	81%	74%	73%	73%	74%
H	76%	69%	61%	59%	74%	82%
Student No.	38	40	36	53	38	39

III. Design the lab manual with engineering learning style

Over the years with the development of computer hardware and software, assembly language for x86 microprocessors has undergone major changes from DOS assembly, 80x86 assembly, Win32 assembly, and Win64 assembly. X86 assembly language mainly has Intel syntax and AT&T syntax. Many schools choose one of them to teach x86 registers and architecture. For our students who are only co-taking C++ programming, we decided to take a different approach to teach basic architecture of x86 processor family by C program and inline assembly language module embedded into a C program. We use the two assembly language syntax branches to teach registers, flags, ALU, memory contents, memory addresses, a stack structure, I/O ports, computer architecture, etc. However, we could not find a good lab manual for a Microcomputer Systems Technology lab class. The course coordinator provided the lab handout in each semester. Students had difficulty following the lab class instruction. We began to contact a higher education publisher in 2017 and designed a lab manual to revise and update some of the previous lab handouts and newly developed exercises. The lab manual designs incorporate the course outline, the assessment blueprint, student learning styles, effective teaching strategies, C/C++programming, x86 assembly language, and our preliminary work [16][17][18][19]. The designed lab manual includes 16 exercises. The contents of lab manual are shown in *TABLE 3*.

TABLE 3
THE TABLE OF CONTENTS OF A LAB MANUAL

PRACTICAL PROGRAMMING EXERCISES USING ASSEMBLY LANGUAGE WITH C/C++	
Exercise 1	Getting Started into x86 Assembly from a C++ Console App Project in Visual Studio
Exercise 2	Data Formats and Data Conversion
Exercise 3	Data Movement between General Purpose Registers
Exercise 4	Memory Addresses
Exercise 5	Addressing Modes
Exercise 6	Extending Signed and Unsigned Numbers
Exercise 7	Integer Arithmetic Operations
Exercise 8	Logic Operations
Exercise 9	Bit Manipulation and Mask Design
Exercise 10	The Stack and LIFO Data Structure
Exercise 11	Processor Flags and Condition Codes
Exercise 12	Floating Point Arithmetic Operations
Exercise 13	Computer Hardware Control Using Input and Output Ports of the PC
Exercise 14	Interfacing to Standard Computer Parallel Ports of the PC – PCI express
Exercise 15	Generating Assembly Code from C code by the GNU Assembler
Exercise 16	Microsoft Visual Studio Community and User Interfaces

Exercise 1 is for getting started into x86 assembly from a C++ console app project in Visual Studio. Students then can follow a similar platform of C++ Console App to work on the first 12 exercises. The method of C/C++ program embedded an inline assembly into a C/C++ program will be used to run practical exercises of the followings: data formats and data conversion, data movement between general purpose registers, memory addresses, addressing modes, extending signed and unsigned numbers, arithmetic operations, logic operations, bit manipulation and mask design, the stack and LIFO data structure, the processor flags and condition codes, and floating-point arithmetic. The exercises of 13, 14, and 15 are designed to run on C/C++ programming via a 64-bit Linux platform. The topics of these three exercises are computer hardware control using input and output ports of the PC, interfacing to standard computer parallel ports of the PC with PCI express, and generating assembly code from C code by the GNU assembler. The Linux I/O port programming and the assembly code generated from a C/C++ program by the GNU assembler will be used to run these three exercises to study the microprocessor's input and output instructions and the registers used in I/O ports. Exercise 16 is designed to bring an opportunity for students to develop an open-source project on the platform of Visual Studio Community. These practical exercises are designed to incorporate with different engineering learning styles based on FSLSM: sensing learning, verbal learning, reflective learning, active learnings, sequential learning, and global learning, as well as the principles of flipped classroom teaching methods in a laboratory course.

This first exercise in the lab manual is designed for a sensing learner. These students can take in information that is realistic and practical towards procedures, details, and figures. The exercises of 2 to 15 are designed for a verbal learner, reflective learner, active learner, and sequential learner. In each lab, two examples are provided. Example 1 shows the relationship between theory and practical solution. It is designed to teach students the approach of learning by doing and just in time teaching the basics. Students are reflective learners since they learn by explaining source code comments of the example. Students are verbal learners since they are required to analyze the output from the executable file and write a lab report for each topic. Example 2 or example 3 provided in each lab topic emphasizes on student-centered active learning experiences with more practical exercises and open-ended questions. Students modified a template example to complete open-ended tasks to reinforce concepts and the understanding of the principle of microprocessor systems and their application. Once students completed a series of practical exercises, they have learned the material in a chronological order by a sequential learning style. They can achieve a level of mastery of the learning goal and learning outcome. Exercise 16 is designed for a global learning style. Students will find the connection to previous ones they have already done and work on open-source projects via a platform of Visual Studio Community. The recorded flipped lab videos will be used as a guideline for students to develop an open source project. Students will watch a video before the laboratory class. With our approach to design a lab manual, we wish it can motivate, coach, and deliver instructions to this diverse student body.

TABLE 4
LOGICAL OPERATIONS AND <bitset> CLASS APPLICATION

```
void Xor_operation1(unsigned short r1, unsigned short r2)
{
    unsigned short r;
    _asm {
      mov AX, r1;
      mov CX, r2;
      xor AX, CX;
      mov r, AX;
   }
    bitset<16> operant1_Bits (r1);
    bitset<16> operant2_Bits (r2);
    bitset<16> result_Bits (r);
    cout << "Perform a XOR operation:" << endl;
    cout <<"\t\t"<< operant1_Bits <<endl;
    cout <<"\tXOR" <<"\t" << operant2_Bits <<endl;
    cout <<"----------------------\n";
    cout <<"\t\t" << result_Bits <<endl;
    cout <<"==================\n";
}
```

```
Perform a XOR operation:
                0001001010101010
        XOR     0001001010101010
----------------------------------
                0000000000000000
==================================
```

```
void Xor_operation2(unsigned short r1, unsigned short r2)
{
    unsigned short r;
    r = r1 ^ r2;
    bitset<16> operant1_Bits (r1);
    bitset<16> operant2_Bits (r2);
    bitset<16> result_Bits (r);
    cout << "Perform a XOR operation:" << endl;
    cout <<"\t\t"<< operant1_Bits <<endl;
    cout <<"\tXOR" <<"\t" << operant2_Bits <<endl;
    cout <<"----------------------\n";
    cout <<"\t\t" << result_Bits <<endl;
    cout <<"==================\n";
}
```

```
Perform a XOR operation:
                0001001010101110
        XOR     1111111100000000
----------------------------------
                1110110110101110
==================================
```

IV. Examples of the Exercises in Lab Manual

The design of microcomputer systems technology lab exercises gives students a better understanding of the register level instructions is our focus. It is difficult to read an assembly code (ASM) and understand the register level instructions from the assembly code generated from the disassembly window. It is also difficult to let students read an ASM generated by a GNU assembler.

We take an approach to the combination of C/C++ console input and output with inline assembly module. For example, the objective of exercise 10, "Logic operations", is to let students write a C/C++ program as well as embedding an inline assembly language module into a C program to exam logic instructions for the processor. To demonstrate bits in the binary format, we use the <bitset> class from C++ language to display the operations of setting bits, clearing bits, and inverting bits. Students work on these exercises to learn C++ class, the function of bitset to convert an integer to binary bits, ASM instructions and C operators of AND (&), OR(|), XOR(^), and NOT(!) used in a mask design for the real engineering solutions. We show an example to write functions in C programming and an inline assembly language module and how to call them inside of the main function (shown in *TABLE 4*).

To understand the registers of AX and DX used in port input and output of the PC, and I/O instructions of IN and OUT, we designed Linux based lab exercises. First, students write a C code with input and output functions inb and outb, and then the AT&T syntax ASM code will be generated by a GNU assembler. For example, the instruction of IN AL, DX will read data at the port address into AL register, where the port address is stored in the DX register. To compare the ASM code with C language functions, a student can understand the connections between C function inb and microprocessor's IN instruction. The *TABLE 5* shows that C language I/O port read function inb(portAddress) is interpreted by ASM. The *TABLE 6* shows the understanding of the connection between write function outb(byteDataSent, portAddress) in C language with the microprocessor's OUT instruction.

TABLE 5
I/O PORT READ FUNCTION

byteDataReceived= inb(portAddress)	

_asm{
mov DX, portAddress;
in AL, DX;
mov byteDataReceived, AL
}
	.type inb, @function
	"/usr/include/x86_64-linux gnu/
	sys/io.h"
	inb %dx, %al

TABLE 6
I/O PORT WRITE FUNCTION

outb(byteDataSent, portAddress)	

_asm{
mov DX, portAddress;
mov AL, byteDataSent;
out DX, AL
}
	.type outb, @function
	"/usr/include/x86_64-linux-gnu/
	sys/io.h"
	outb %al, %dx

V. Conclusion

The lab manual design with engineering learning style and flipped learning model in Microcomputer Systems Technology course has been discussed. It incorporates the course outline, assessment blueprint, student learning style, effective teaching strategie, C/C++programming, x86 assembly language, and our preliminary work. Sixteen weekly lab exercises have been developed to teach students to master concepts and enhance their ability for creative application. At the end of the semester, students should be able to work on the platform using multiple programming languages (C/C++ and Assembly), and multiple hardware devices (PC or Laptop, x86 device, Raspberry Pi) on different operating systems (MS Windows, Linux). The student performance SLOs of A, B, C, D, E, F, G, and H will continue to be evaluated every fall semester based on an assessment blueprint.

References

[1] H. Pashler, M. McDaniel, D. Rohrer, R. Bjork, "Learning Styles Concepts and Evidence," *Psychological a journal of scientific in the public interest (SAGE)*, vol. 9, issue 3, pp. 105-119, 2008

[2] R.M., Felder, & L.K. Silverman, "Learning and teaching styles in engineering education," *Engineering Education*, vol. 78, no.7, pp. 674-681, 1988

[3] R.M. Felder,, & J. Spurlin, "Applications, reliability, and validity of the index of learning styles," [Electronic Version]. *Int. J. Engng Ed*. vol. 21, no.1, pp.103-112, 2005

[4] Y. Kim and C. Ahn, "Effect of Combined Use of Flipped Learning and Inquiry-Based Learning on a System Modeling and Control Course," *IEEE Transactions on Education*, vol. PP, no. 99, pp. 1-7, Dec. 2017

[5] G. S. Mason, T. R. Shuman, and K. E. Cook, "Comparing the effectiveness of an inverted classroom to a traditional classroom in an upper-division engineering course," *IEEE Trans. Educ*. vol. 56, no. 4, pp. 430–435, Nov. 2013

[6] J.L. Bishop, M.A. Verleger, The flipped classroom: A survey of the research, in *Proc. of the 120th ASEE annual conference & exposition*, Atlanta, GA, June 23-26, 2013.

[7] Aliye Karabulut-Ilgu, Suhan Yao, Peter Savolainen, Charles Jahren, "Student Perspectives on the Flipped-Classroom Approach and Collaborative Problem-Solving Process," *Journal of Educational Computing Research*, Aug 23, 2017

[8] Joseph Ranalli, Jacob Moore, "Targeted flipped classroom technique applied to a challenging topic," *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1 - 4, 2016

[9] Ricky T. Castles. "Development of Leadership through Hands-On Learning Activities in a Flipped Microprocessors Classroom," in *Proc. of 2017 ASEE Annual Conference & Exposition*, 2017

[10] Edward F. Gehringer, Barry W. Peddycord, III, "The inverted-lecture model: a case study in computer architecture," *SIGCSE '13: Proc. of the 44th ACM technical symposium on Computer science education*, pp. 489-494, 2013

[11] Redekopp, M.W., Ragusa, G. "Evaluating Flipped Classroom Strategies and Tools for Computer Engineering," in *Proc. of the 120th ASEE annual conference & exposition*, Atlanta, GA, June 23-26, 2013

[12] [Online]. Available: http://www.abet.org/

[13] Eip R. Irvine, *Assembly Language for x86 Processors*, 7/E, Pearson, 2015

[14] Barry B. Brey, *Intel Microprocessors*, *8/E*, Pearson, 2009

[15] Hyde, R., *The Art of Assembly Language*, No Starch Press, 2010

[16] Yu Wang, Alex Wong, and Aparicio Carranza, "The Course Development for Microcomputer Systems Technology: Preliminary Study," *Proc. of 2013 ASEE Mid-Atlantic Spring Conference*, New York, April 26-27, 2013

[17] Yu Wang, Farrukh Zia, Ohbong Kwon, and Xiaohai Li, "Collaborative Instruction and Team Based Project Learning - An Effective Strategy to Conduct Technology Education," *Proc. of 2015 ASEE Northeast Conference*, Boston, April 30-May 2, 2015

[18] José M. Reyes Álamo, Yu Wang, and Renata Budny. "Bridging the Gap between General Education and Accredited Engineering Technology Fields", *Proc. of 2017 ASEE Middle Atlantic Spring Conference*, Baltimore, Maryland, April 7-8, 2017

[19] Andy S. Zhang, Angran Xiao, Yu Wang, Farrukh Zia, and Muhammad Ummy, "A Hands-on Robotics Concentration Curricula in Engineering Technology Programs," *Proc. of 2018 Conference for Industry and Education Collaboration,* San Antonio, TX, February 7-9, 2018