Cost-efficient Hands-on Learning Design for Computer Organization Course

Suk Jin Lee
TSYS School of Computer Science
Columbus State University
Columbus, GA, USA
lee_suk@columbusState.edu

Andrew Jung
Dept. of Computing Sciences
University of Hartford
West Hartford, CT, USA
jung@hartford.edu

Jinsook Park
Dept. of Mathematics
University of Hartford
West Hartford, CT, USA
jpark@hartford.edu

Mira Yun
Dept. of Computer Science & Networking
Wentworth Institute of Technology
Boston, MA, USA
yunm@wit.edu

***Abstract*— This paper has probed an innovative hands-on activity for Computer Organization course with a cost-effective laboratory setting. A Computer Organization is one of the most important subjects for computer science (CS) students, because the subject focuses on fundamental relationship between hardware and software components in computer systems. Due to its importance, effective and appropriate pedagogies are required in this area. Simulation tools were widely used to visualize the processing of instructions and help understanding the concept of computer systems rather than teaching the concept with traditional lecture method. Teaching method combined with simulators has been issued on the table and raised an inquiry whether it really relates to meaningful student involvement or not. The use of hands-on activities has been shown to improve students' interest in and ability to understand course material. We designed a cost-effective laboratory setup and a set of hands-on activities for the course using low-cost single board computers, i.e. Raspberry Pi. This article introduces how to combine hardware components (e.g. wires, LEDs, resisters, breadboard, ICs) and Python programs to develop hands-on activities for the Computer Organization course. The survey results including fall 2018, spring 2019 and fall 2019 show that almost 90% of students prefer to learn through hands-on activities and the activities during the class helped improve their learning.**

Keywords—Computer Organization, Instructional Approach, Raspberry Pi and Python, Hands-on Activities

I. INTRODUCTION

Computer Organization and Architecture is one of the most important subjects for computer science (CS) students, because the subject focuses on fundamental relationship between hardware and software components in computer systems [1][2]. It is very important for CS students to understand internal working, implementation, and architecture of a computer systems in order to design and develop new services and applications. Since most of the undergraduate institutes require Computer Organization or Computer Architecture for CS degree, there is a strong need for effective and appropriate pedagogies.

Simulation tools were widely used to visualize the processing of instructions and help understanding the concept of computer systems rather than teaching the concept with traditional lecture method [3][4]. As the computer hardware systems become more complicated, the instructors take more simulation tools for pedagogical variation. However, using simulators for hardware interactions has been issued on the table and raised an inquiry on student engagement in the classroom.

The use of hands-on activities has been shown to improve students' engagement and ability to understand course material [5]. In addition, the students can acquire a clear picture of what to expect after having hands-on activities. However, it is very challenging to set up equipment or devices in the classroom without a great amount of financial support. The purpose of this study is to design a cost-effective laboratory setup using low-cost single board computers, i.e. Raspberry Pi, and provide students with a set of hands-on exercises for Computer Organization. One of the most powerful features of the Raspberry Pi is the row of general-purpose input/output (GPIO) pins along the top edge of the board. These pins allow students to connect Raspberry Pi to a range of devices, from LEDs, Integrated Circuits (ICs) and a variety of sensors. This GPIOs can be used as the interface to control tangible tools with Python programming language. This paper presents how to combine hardware components (e.g. wires, LEDs, resisters, breadboard, ICs) and Python programs for undergraduate Computer Organization course. After students perceive the theoretical knowledge of each topic, they can engage in practical exercises. Among the topics covered in the activities are the following: 1) Blinking LEDs, 2) AND Logic Gate, 3) Decoder, 4) Full Adder, 5) SR Latch, 6) Latch and Output Buffer, 7) Clock and Counter, 8) Counter and Decoder, 9) Arithmetic Unit, Latch and Output Buffer, and 10) ARM assembly program. We used various tangible tools for undergraduate students to be more engaged in learning and to make sense of the concepts.

The rest of this paper organized as follows. Section II present the motivation and related works. Section III provides the details about the course contents including the course design, 10 different hands-on activities, and example student works. In Section IV, we share our survey results. Finally, we conclude our work in Section V.

II. MOTIVATION AND RELATED WORKS

Students are keen to learn by doing with real tools rather than perceiving theoretical concepts with pen and paper. Computer Architecture and Organization courses provide students with the knowledge of basic architecture and functions of computer system including data representation, Boolean algebra and logic gates, combinational and sequential circuits, computer arithmetic, instruction set architecture, memory hierarchy and interaction of machine and computer languages. The paper-based methodology appears to be

monotonous and no longer fashionable to undergraduates [6]. The lack of practical work and hands-on activities disturbs students from understanding the theoretical aspect of internal structure of computer systems [7].

CS students benefit from the hardware experience that can establish a clear picture of relevant relationship among a great number of hardware components [8]. The hands-on activities can provide practical experience of hardware with CS students [9][10]. Through hands-on activities the students can quickly grasp the concepts of computer operations, examine the CPU structure while executing program, and acquire the ability to adjust expert skill in their careers [10]. The practical activity can also make them enjoy learning [11].

For Computer Architecture and Organization, simulators can easily visualize the processing of instructions. Grunbacher [12] developed several pipeline simulators and a cache simulator based on Hennessy & Patterson's architecture and MIPS processor book. Atanasovski et al. [13] also elaborated EDUCache simulator as a supporting tool for students' understanding the concepts of computer architecture and organization. They developed a platform independent simulator using Java. Their simulator focus on designing modern multi-layer, multi-cache and multi-processors. They further extended the simulator by visualizing the cache behavior dependent on various cache parameters [14]. Recently Lopez-Rosenfeld [6] demonstrated that simulators combined with hands-on course model help students understand how the arithmetic logics operate.

As computer system and hardware gets more complex, the computer architecture instructors adopt more tools for pedagogical and/or research purposes. Many free simulators are available on the Internet contributing as helpful and practical resources for instructors as well as students since late nineties [15][16]. However, these researches only focused on introducing available simulators and their features to promote students' understanding. Chen et al. [17] designed different teaching methods to improve teaching effect and promote students' learning experience. The whole course includes theory teaching, the training of practice methods and hands-on labs. The students are able to learn how to prepare and perform the hardware component and the computer systems through FPGA (Field-Programmable Gate Array) and Verilog language. Gao et al. [18] designed an analogous teaching method by combining simulators with FPGA platforms. Lo et al. [4] proposed to use a low-cost portable microcontroller, MSP 430 manufactured by Texas Instrument Inc. This simple portable device makes it possible to execute simple projects for fundamental computer systems, but their learning module is limited to MSP430 assembly programming and simulator.

"Learn-by-doing" teaching paradigm provides students with a motive to engage in their learning. Although the simulators have been widely used as affordable and easy access resources for Computer Organization, we believe that various tangible tools can improve the engagement and learning experience of undergraduates. We designed a cost-effective laboratory setup and a set of hands-on activities for the course using low-cost single board computers, i.e. Raspberry Pi. By combining various hardware components (e.g. wires, LEDs, resisters, breadboard, ICs) and Python programs, we provide students strong visualizations and hands-on experience on the interactions between hardware and software components of computing systems.

III. HANDS-ON ACTIVITIES FOR COMPUTER ORGANIZATION

A. Course Information

The Computer Organization course introduces the foundation of computer design, implementation, and operations. This course provides basic concepts in digital logic circuits, computer arithmetic, computer microarchitecture, and memory hierarchy. At the completion of this course, the student should be able to demonstrate a competence of basic skills in digital logic analysis and design, computer arithmetic, computer instruction set architectures optimization, assembly language programming, computer microarchitecture, and memory hierarchy.

Table 1 shows the weekly schedule for lecture topics and hands-on activities. For the first three weeks of the course, the students are engaged in understanding the basic computing hardware, logic gates, and Boolean expression. In the third week, the basic concept of Raspberry Pi is introduced. This hands-on lab (*00_Intro-to-RPi*) is quite important to students because it delivers Raspberry Pi hardware specifications, useful tools for Raspberry Pi, GPIO pin settings, and some handy Linux commands. In Week 4, the students start to analyze combinational circuits, including a binary encoder & decoder, multiplexers, 1-bit adders, and a priority circuit. The *01_Blinking-LED* hands-on activity instructs students to design a simple circuit with resistor, wire, and breadboard to turn LED on/off using Python programming. In Week 5, students learn how to represent both integer and fraction numbers into binary numbers in fixed-point/floating-point number formats. Since the course schedule introduces the basic logic gates in Week 2, the students can exercise how to use basic logic gates (AND gate) in *02_Logic-Gate* activity. In Weeks 6, 7 and 9, lecture topics are extended to computer arithmetic, combinational circuit and sequential circuit designs with decoder, full adder, and SR latch hands-on activities. Week 10 gives students a bird's eye view of CPU. *06_Latch-n-Output_Buffer* activity emulates designing shared bus with Latches and 3-state Bus Buffers. Week 11-Instruction Set Architecture covers different instruction formats and their interpretation to machine language with program counter (PC) activity. After the students are aware of logic and conditional operations of assembly languages through Week 12 – 14 activities, students can write and test ARM assembly program to execute arithmetic operations.

TABLE I. WEEKLY SCHEDULE

Week	Lecture Topics	Hands-on Activities
1	Class Administration Overview on Computing Hardware	-
2	Basic Hardware Building Blocks	-
3	Boolean Expressions	*00_Intro-to-Rpi*
4	Combinational Circuit Analysis	*01_Blinking-LED*
5	Binary Number Formats	*02_Logic-Gate*
6	Computer Arithmetic	*03_Decoder*
7	Combinational Circuit Design I	*04_Full-Adder*
8	Midterm Exams	-
9	Combinational Circuit Design II	*05_SR-Latch*
10	Basic CPU Organization	*06_Latch-n-Output_Buffer*
11	Instruction Set Architecture	*07_Clock-n-Counter*
12	Assembly Languages	*08_Counter-n-Decoder*
13	Instruction Pipelining	*09_Arithmetic Unit, Latch and Output Buffer*
14	Memory Hierarchy	*10_Assembly program*
15	Memory Addressing	-
16	Final Exam	-

B. *Hands-on Activities*

1) *Raspberry Pi and Python*

00_Intro-to-Rpi activity introduces the basic concept of the Raspberry Pi and helps the students install the pigpio library, where the activity requires an Internet connection to install the library. The students also test the GPIO pins and practice some handy Linux commands that run at Raspbian cmd prompt commands.

The students start to use pigpio to control devices (LEDs) in *01_Blinking-LED* activity. They are instructed to design a simple circuit with an LED and a resistor, and then write and run a Python program to switch LED on/off on Raspberry Pi. The students should get the detailed instruction to design the circuit in the step, e.g. how to use a breadboard and the positive and negative sides of LED. As shown in Fig. 1, a sample Python program is given to test the setup, so that the students can learn how to control the device with Python program.

```
import RPi.GPIO as GPIO          # Import the RPi.GPIO library
import time                      # Import the Time library
GPIO.setmode(GPIO.BCM)           #  BCM:  Broadcom  SOC  channel,  cf.
BOARD
GPIO.setwarnings(False)          # Do not print GPIO warning messages
LED = 4                          # Use BCM pin number 4
GPIO.setup(LED,GPIO.OUT)
print "Switching LED on"         # Print some info to the terminal
GPIO.output(LED,GPIO.HIGH)       # GPIO.HIGH = 1
time.sleep(2)                    # Pause the Python program for 2 sec
print "Switching LED off"        # Print some info to the terminal
GPIO.output(LED,GPIO.LOW)        # GPIO.LOW = 0
GPIO.cleanup()
```

Fig. 1. Python program to test blinking LED

After their practical experience of switching LED on/off, they are requested to extend their knowledge to solve a real-world problem. The challenge is to develop a Python program to simulate traffic lights with an instruction. In order to complete the challenge, the students should connect a red LED to GPIO pin 12, a yellow LED to GPIO pin 16, and a green LED to GPIO pin 21. This activity helps students become aware of real-world problems and can be solved by writing computer programs

2) *Designing Logic Circuits*

In *02_Logic-Gate* activity, students design a logic circuit with AND gate (SN74LS08N) and develop a Python program to test the operation of the logic gate, as shown in Fig. 2. This hands-on lab helps students' understanding of controlling logic AND gate and LEDs. After designing the logic circuits with an AND logic gate, the students are instructed to develop a Python program to simulate the operation of AND logic gate.

For the understanding of combinational circuits, students design a logic circuit with 4-line BCD to 10-line decimal decoder (SN74LS42N) in *03_Decoder* activity, as shown in Fig. 3. Using a Python program, students can convert the decimal number into the binary inputs which will be fed into

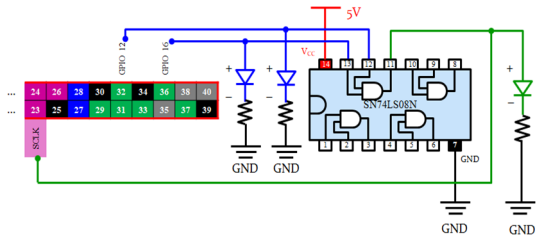

Fig. 2. Circuit design with a AND logical gate

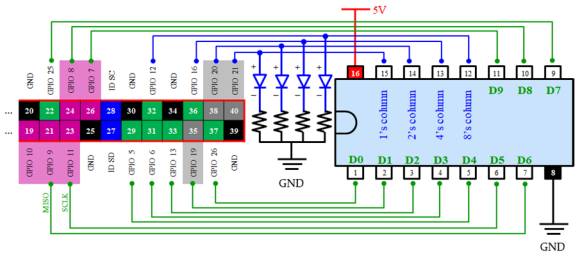

Fig. 3. Circuit design with 4-Line BCD to 10-Line Decimal Decoder.

the binary inputs of SN74LS42N and test the operation of 4-line BCD to 10-line decimal decoder. The binary inputs are visualized with four LEDs, where each one has a different weight. This hands-on Lab evaluates students' practical knowledge of converting decimal to binary and controlling 4-to-10 decimal decoder.

A full adder is one of arithmetic devices in CPU. Students design a logic circuit with 4-bit binary full adder (SN74LS283N) to understand arithmetic operations in *04_Full-Adder* activity, as shown in Fig. 4. Students are instructed to develop a Python program to convert the decimal number into the binary inputs and execute adder operations using the designed circuit.

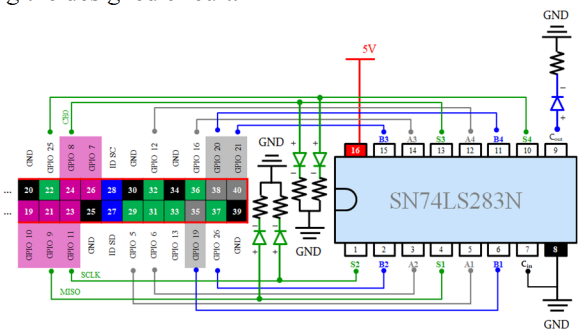

Fig. 4. Circuit design with 4-Bit Binary Full Adder

In *05_SR-Latch* activity, students design a sequential logic circuit with NOR gate (SN74LS02N), as shown in Fig. 5, and then learn how to store one-bit state (0 or 1) by developing a Python program to simulate the operation of the circuit. LEDs attached to IC pins also visualized the outputs, i.e. Set, Reset, and Q values in the circuits.

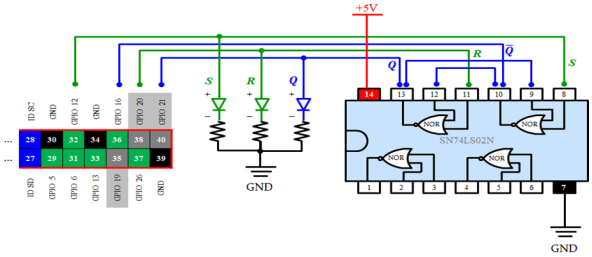

Fig. 5. Sequential circuit design with NOR gate.

3) *Designing ALU and processor*

One of the learning objectives in this course is to understand the basic concepts of sequential circuits and storage elements in Arithmetic Logic Unit (ALU). This can relate to generate binary inputs using the Python program, store the binary inputs into clock-controlled latches and forward the stored values using 3-state bus buffers. In the hands-on Activity *06_Latch-n-Output_Buffer*, students design clock-controlled latches with 4-Bit Bistable Latches

(SN74LS75N) and 3-state Bus Buffers (SN74LS126N), as shown in Fig. 6. This activity helps students understand how the stored binary inputs in the latches are forwarded to the shared bus with the 3-state Bus Buffers. The students should develop a Python program to create clock-controlled latches with an instruction of sequences to test the circuit operation. This hands-on Lab evaluates students' practical knowledge of converting decimal to binary, storing the binary values with clock-controlled latches and controlling 3-state Bus Buffers.

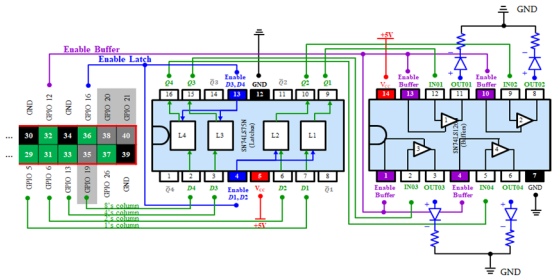

Fig. 6. Circuit design with 4-bit Bistable Latches and 3-state bus buffers.

A program counter (PC) is a register in a computer processor that contains the address (location) of the instruction being executed at the current time. The activity *07_Clock-n-Counter* helps students design a binary counter circuit to emulate PC with binary counters (SN74LS90N), as shown in Fig. 7. Students generate clock signals for the binary counter, a periodic signal to repeat logic High and Low with a Python program. The generated clock signals can be visualized with a LED connected to GPIO pin 12 in Fig. 7.

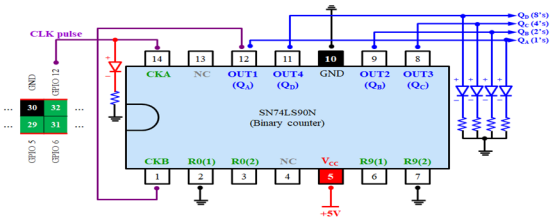

Fig. 7. Circuit design with a binary counter.

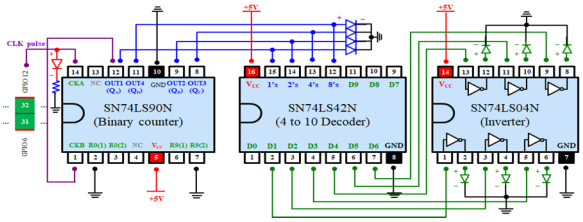

Fig. 8. Circuit design with a binary counter and 4-to-10 Decoder.

The activity *08_Counter-n-Decoder* extends the activity *07_Clock-n-Counter* by adding 4-to-10 Decoder and Inverter, as shown in Fig. 8. With this activity, students develop a Python program to generate clock signals, count the binary signal with a binary counter (SN74LS90N) and represent 4 bits binary number into decimal number using 4-to-10 Decoder (SN74LS42N). Due to the limited pin numbers of Inverter (SN74LS04N), the decimal number only counts one through six in the given circuit.

The activity *09_Arithmetic_Unit_Latch_Output_Buffer* helps students understand how a CPU executes the whole process with control signals (enable/disable); perform an (arithmetic) addition using binary full adders (SN74LS283N), store the binary results into Latches (SN74LS75N) and forward the stored values using 3-state Bus Buffers (SN74LS126N), as shown in Fig. 9. Due to the complex circuit configuration, the detailed instructions should be provided to help students complete their task successfully.

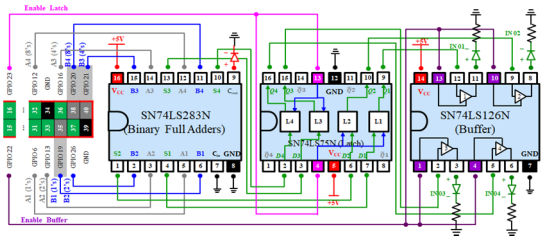

Fig. 9. Circuit design with with a full adder, Latches and 3-state Buffers.

In the activity *10_Assembly_program*, students write and test ARM assembly program language to execute arithmetic operations. For example, the high-level language program is given to calculate the power operation. Students should write an ARM assembly programming language that is equivalent to the high-level language program.

C. Student Work

1) Raspberry Pi and Python

The hands-on activity *01_Blinking-LED* provides students with the first challenge to design a simple circuit with real hardware and to solve a real-world problem. Fig. 10 is one of the most students' solution of *01_Blinking-LED* activity.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)  #setting up the program and LED's
GPIO.setwarnings(False)
R_LED = 12
Y_LED = 16
G_LED = 21
GPIO.setup(R_LED,GPIO.OUT)
GPIO.setup(Y_LED,GPIO.OUT)
GPIO.setup(G_LED,GPIO.OUT)
while(True):
    GPIO.output(R_LED,GPIO.HIGH)
    time.sleep(1)
    GPIO.output(R_LED,GPIO.LOW)  #Turn on the red and yellow LED's
    GPIO.output(R_LED,GPIO.HIGH)
    GPIO.output(Y_LED,GPIO.HIGH)
    time.sleep(1)
    GPIO.output(G_LED,GPIO.HIGH) #Turn on the green LED
    time.sleep(4)
    GPIO.output(G_LED,GPIO.HIGH) #Turn on the green and yellow LED
    GPIO.output(Y_LED,GPIO.HIGH)
    time.sleep(3)
    GPIO.output(R_LED,GPIO.HIGH) #Turn on the red LED
    time.sleep(1)
    GPIO.output(R_LED,GPIO.LOW)  #Turn all of the LED's off
    GPIO.output(Y_LED,GPIO.LOW)
    GPIO.output(G_LED,GPIO.LOW)
GPIO.cleanup
```

Fig. 10. Example 1 of students' work for *01 Blinking LED*.

2) Designing Logic Circuits

The hands-on activity *04_Full-Adder* provides students with a deep understanding of arithmetic operations with 4-bit binary full adder (SN74LS283N). The challenge of the lab is to design a subtractor, by connecting IC pin 7 (cin) from GND to +5V for the hardware side and flipping all the bits of Input B with 'not' logic operator in the software side. We observed that some students experienced difficulty in designing the subtractor, as shown below with some comments:

"*This activity taught me a lot on how to work a circuit board. It was very difficult at the beginning in order to find out how the subtractor worked. Having to do the adder was very easy to understand but finding out how to do the subtractor was very difficult. I learned a lot during the process in order to find the subtractor. My part to do on this project*

was to check to see if there was anything wrong with the code and make sure the bread board worked accordingly. This project really taught me a lot."

"*throughout this activity I learned that circuit boards and coding are a very difficult concept that takes a lot of patience and knowledge on how to do it. Also, I learned that changing to ground to +5v makes the circuit a 'subtractor' instead of the 'adder'. My role in this project was writing the code, and I also tried to understand how the circuit board actually worked which was difficult. The code completely runs the wiring had a few problems that might have been caused by the misplacement of wires that I could not find since most of the program ran correctly in correlation with the circuit board.*"

3) Designing ALU and processor

The hands-on activity *09_Arithmetic_Unit_Latch_Output_Buffer* provides students with deep understanding of processor architectures. The challenge is not only to design the complex circuit configuration but also to require deep understanding of overall system architectures to execute arithmetic operation and generate the enable signals for latches and buffers. We observed that the students should be able to complete the challenge by retrieving previous hands-on activities, as follows below with some comments:

"*For this go around we got to use our prior knowledge and convert the decimal number into the binary inputs. We learned how to perform addition using binary full adders. And then with the addition, store the binary results into Latches and forward the stored values using 3-state Bus Buffers. Student_1 did what he did best, and built our circuit. While Student_2 worked on making sure that the program would be ready to run when the circuit was built. We have become really efficient in making sure that everything gets done as quickly and as accurately as possible. Of course, we always make sure that the other one gets any kind of input they may need and we make sure to help or give ideas to each other.*"

IV. RESULTS

We took the surveys for three consecutive academic semesters from 2018 Fall through 2019 Fall. The total numbers of participants for the pre-survey were 27 students for Fall 2018, and 26 students for both Spring 2019 and Fall 2019. The total numbers of participants for the post-survey were 25 students for Fall 2018 semester, 24 students for Spring 2019 semester, and 16 students for Fall 2019 semester. Students participate in both surveys voluntarily and some students withdraw the course in the middle of the semester.

The pre-survey was taken at the beginning of the semester, and the post-survey was performed at the end of the semester. Both pre and post survey questions use Linkert scales. The result of the pre-survey showed that students during the three consecutive semesters have been motivated to learn computer hardware system at the beginning of the semester. Majority of the students responded to their interests in computer hardware system, as follows: 81.4% of students for Fall 2018, 76.90% of students for Spring 2019, and 61.50% of students for Fall 2019. Fig. 11 shows the number of students answered for each scale. Fig. 12 shows the results of the question *"I think this course will be useful for my future career."*. 88.88% of students in Fall 2018, 92.30% of students in Spring 2019, and 61.60% of students in Fall 2019 answered "Agree" or "Strongly Agree" for this question. Students recognized the importance of the course for their future career.

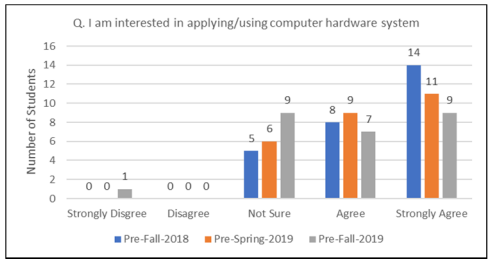

Fig. 11. Students' Interest in Computer Hardware System.

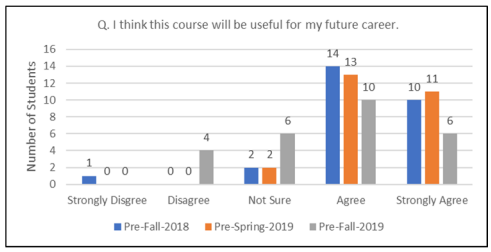

Fig. 12. Usefulness of the course for the career preparation.

Fig. 13 represents the results of hardware computing problem solving ability of Fall 2018, Spring 2019 and Fall 2019. 34.61% of students in Fall 2018, 46.1% of students in Spring 2019, and 46.1% students in Fall 2019 answered that they have hardware-computing program solving ability from pre-survey. However, in post-survey, 60% of students in Fall 2018, 62.8% of students in Spring 2019, and 62.5% of students in Fall 2019 answered either "Agree" or "Strongly Agree". We believe that this increase is caused by 10 hands-on activities with tangible object (Raspberry Pi) combined with Phyton programming challenges. As shown in Fig. 14, 52% of students in Fall 2018, 75% of students in Spring 2019, and 81.30% of students in Fall 2019 revealed that they like to do Raspberry Pi activities.

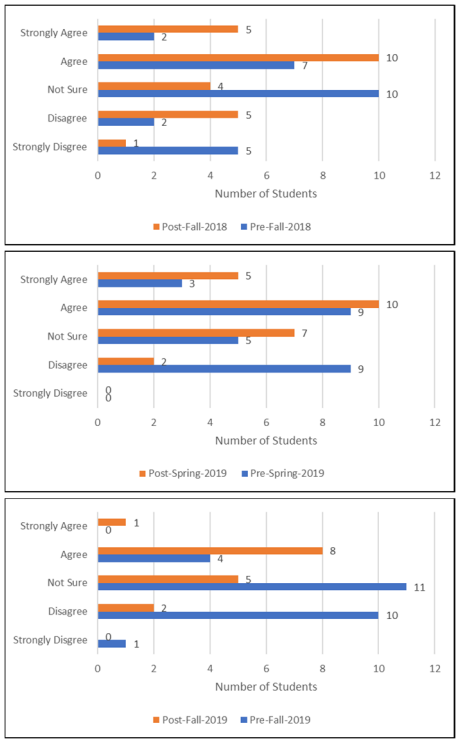

Fig. 13. Self-efficacy of Hardware Computing Problem Solving Capability for Fall 2018, Spring 2019 and Fall 2019 semesters.

The survey also indicated that the activities performed in the class enhanced students' learning experience. Fig. 15 showed that 76% of students in Fall 2018, 87.50% of students in Spring 2019, and 81.10% of students in Fall 2019 responded with either "Agree" or "Strongly Agree" for the usefulness of activities to enhance learning.

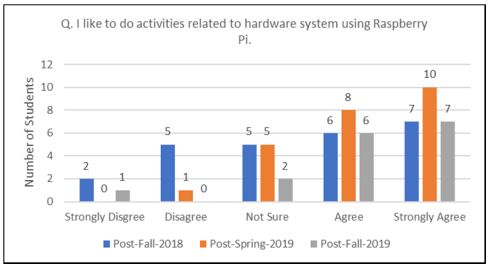

Fig. 14. Students' preference of activities using Raspberry Pi.

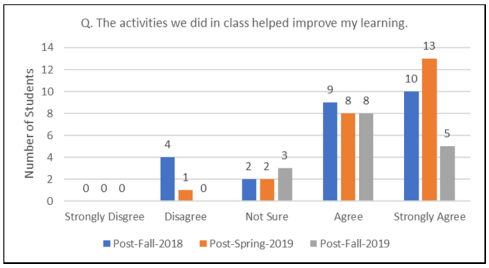

Fig. 15. Advantage of class activities for students' learning.

As shown in Fig. 16, above 76% (up to 96% in Spring 2019) students agree that this course helped students to understand what hardware computing system looks like.

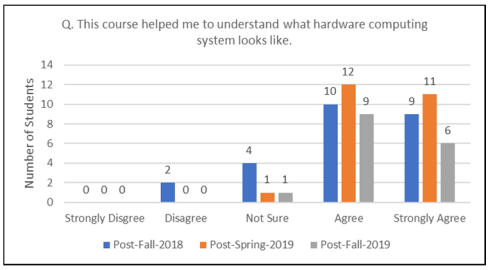

Fig. 16. Usefulness of the course for students' learning about hardware computing system.

V. Conclusion and Future Work

In this paper, we introduced a cost-effective laboratory setup using Raspberry Pi and made it available a set of hands-on exercises for Computer Organization. By sharing 10 different Raspberry Pi and Python programing activities, we provide students strong visualizations and hands-on experience on the interactions between hardware and software components of computing systems. The survey results showed that students are engaged in the course by using the hands-on activities, but also showed that hardware computing problems are likely difficult to CS students who directly dived into computer hardware systems without any knowledge of computer hardware. We are planning to extend the designed laboratory setup to other curriculum models to show the efficacy of hands-on activities.

References

[1] Shine V.J. et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (2) , 2014, pp. 1411-1413.

[2] David A. Patterson and John L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 5th ed., 2013.

[3] Wijaya Kurniawan and Mochammad Hannats Hanafi Ichsan, "Teaching and learning support for computer architecture and organization courses design on computer engineering and computer science for undergraduate: A review," 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Sept. 19-21, 2017.

[4] Dan Chia-Tien Lo, Kai Qian, Liang Hong, "The use of low cost portable microcontrollers in teaching undergraduate Computer Architecture and Organization," IEEE 2nd Integrated STEM Education Conference, 2012.

[5] Medaris, K., "Study: Hands-on projects may be best way to teach engineering and technology concepts," 28 January 2009. [Online]. Available: https://news.uns.purdue.edu/x/2009a/090128DarkStudy.html [Accessed 13 May 2020].

[6] Matias Lopez-Rosenfeld, ""Tell Me and I Forget, Teach Me and I May Remember, Involve Me and I Learn": Changing the Approach of Teaching Computer Organization," 2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM), 2017.

[7] Osama Ahmed Siddiqui, Raza Hasan, Salman Mahmood, Asim Rasheed Khan, "Simulators as a Teaching Aid for Computer Architecture and Organization," 2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics, 2012.

[8] Bo Hatfield, and Lan Jin, "Improving Learning Effectiveness with Hands-On Design Labs and Course Projects for the Operating Model of a Pipelined Processor," 2010 IEEE Frontiers in Education Conference (FIE), 2010.

[9] N.L.V. Calazans, F.G. Moraes, and C.A.M. Marcon, "Teaching computer organization and architecture with hands-on experience," 32nd Annual Frontiers in Education, 2002.

[10] Ian McLoughlin and Koji Nakano, "A Perspective on the Experiential Learning of Computer Architecture," 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, 2010.

[11] Jihane Kojmane and Ahmed Aboutajeddine, "Enjoyeering Junior: A hands-on activity to enhance technological learning in an engineering dynamics course," 2016 International Conference on Information Technology for Organizations Development (IT4OD), 2016.

[12] Herbert Grunbacher, "Teaching computer architecture/organisation using simulators," FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. Conference Proceedings (Cat. No.98CH36214)," vol 3, 1998.

[13] Blagoj Atanasovski, Sasko Ristov, Marjan Gusev, Nenad Anchev, "EDUCache simulator for teaching computer architecture and organization," 2013 IEEE Global Engineering Education Conference (EDUCON), 2013.

[14] Marjan Gusev, Sasko Ristov, Dimitrij Mijoski, "Enhancing the EDUCache simulator with visualization of cache performance," 2016 IEEE Global Engineering Education Conference (EDUCON), 2016.

[15] Wolffe, G. S., Allendale, M. I., Yurcik, W., Normal, I. L., Osborne, H., UK, W. Y., ... & CuIIowhee, N. C. (2002). Teaching Computer Organization/Architecture With Limited Resources Using Simulators.

[16] Prasad, P. W. C., Alsadoon, A., Beg, A., & Chan, A. (2016). Using simulators for teaching computer organization and architecture. Computer Applications in Engineering Education, 24(2), 215-224.

[17] Tianzhou Chen, Guanjun Jiang, Wei Hu, Xueqing Lou, "The Innovation and Reformation of Teaching Method for Computer Organization and Design Course," 2009 International Conference on Information Engineering and Computer Science, 2009.

[18] Zhigang Gao, Huijuan Lu, Hongyi Guo, Yanjun Luo, Yunfeng Xie, Qiming Fang, "An Analogous Teaching Method for Computer Organization Course Design," 2016 8th International Conference on Information Technology in Medicine and Education (ITME), 2016.