Does Badge-Based Learning Buck the Grading Curve? An Educational Experiment in Computer Architecture

Peter Jamieson (jamiespa@miamioh.edu)
Dept. of Electrical and Computer Engineering
Miami University

Abstract—*In this paper, we look at how a course implemented as a traditional lecture and as a badge-based course compare in terms of grade distribution. Our pilot badge-based course was taught in 2013 and 2014, and our grade results show that the grade distribution was similar to the lecture-based course with the added benefit of students at the upper end of the class demonstrating significantly deeper knowledge in the subject matter. The low-performing students, arguably, learned the same amount of material as they would have in a lecture based course, and benefited from not wasting their own time (and others) in participating in things they were not interested in. Similarly, more ambitious students were able to pursue interest paths within computer architecture at their own pace.*

Keywords: Badges, Grades, Computer Architecture

1. Introduction

In this paper, we look at a computer architecture course implemented as a traditional lecture and as a badge-based course. The key question we have with the non-traditional badge-based course is would the grade distribution be significantly different than a lecture based course? In a worse case scenario, we might find that all students fail the badge-based course or, oppositely, all receive an excellent grade, which does not, normally, happen in our lecture based course taught two times previously.

A badge-based course provides the learner with a set of badges that can be worked on and demonstrated to provide evidence of mastery of a specific set of skills. In our course, badges are distinguished in three categories: basic, intermediate, and advanced. There are three basic badges that must be completed to get a passing grade in the course. Completion of additional badges maps a student's grade to higher and higher letter grades in the course. The motivation behind badge-based learning for this course is to allow people with high interest to delve deeper into the material while still providing less interested students a basic knowledge. Also, badge-based learning is assessed via deliverables and instructor interviews and can be redone if not satisfied. This assessment differs from tests and assignments, which are one off assessment points that can go poorly independent of a students learned skills.

We have taught the lecture based architecture course two times before teaching the course with our badge-base approach in 2010 and 2011. The class size was approximately 40 for the lecture based courses and 10 and 20 for the badge-based course, respectively. The grade distribution was similar for all courses taking into account the small population for the badge-based course. Note, that the results for the 2014 are still in progress. We are excited with the results with this preliminary study since during the badge-based course, we noticed that interested students delved significantly deeper into computer architecture, and less motivated students learned the same material as they would have in a lecture based course.

2. Background

Badge-based learning hinges on ideas of project-based learning [1], experiential learning [2], and formative assessment [3], [4] among others. Recently, there has been renewed attention on badge-based learning partly because of the competitions sponsored by the Mozilla Open Badges Framework [5] and the Digital Media + Competition [6]. Because of these new activities a number of studies ([7], [8], [9], [10]), innovations ([11]), and articles ([12], [13], [14]) have emerged that deal with this learning approach.

The idea of a merit based badge evolved from the British army, and today badges are common among organizations such as girl and boy scouts among many other groups. The base idea of a merit badge is that the badge is awarded once an individual completes a set of tasks, and in an educational setting these tasks demonstrate mastery of a particular topic. The digital badge, which is a new term, is a badge that is awarded and stored on-line as opposed to made, given, and displayed on a uniform. Gibson *et. al.* article [14] provides a good introduction to this emerging educational paradigm.

Even though there has been increased research activity in badge-based learning, there is not a significant body to leverage in this domain. This is the first, to our knowledge, paper that compares the assessment impact of badge-based learning to the traditional grade distribution of a course. We, however, do not adhere to any need for matching grade distributions between the lecture and badge-based versions of the course. Anecdotally, we believe that students in the badge-based learning approach learned much more than those in a traditional lecture setting though we do not attempt to quantify this.

The hope for badge based learning is to change the motivational structure and assessment of mastery, but however, these reward based mechanisms can themselves have problems; a good summary of some of the concerns can be found in Kohn's book "Punished by Rewards" [15]. This paper is not proposing that badge based learning is the only methodology to teach computer architecture and instead is experimenting with one small piece of the puzzle.

3. Course Organization

In this section, we describe the material taught in computer architecture and detail how the two courses are organized. We make some attempt on quantifying how the two courses compare in terms of deliverables.

3.1 Course Learning Outcomes

At Miami University, computer architecture is taught in the second year of a computer science and computer engineering degree and covers basic computer architecture and the associated assembly language. This material aligns well with two common textbooks used to teach undergraduate students computer architecture; Patt and Patel's, ''Introduction to Computing Systems - from bits & gates to C & beyond" [16], and Patterson and Hennessy's, ''Computer Organization and Design - The Hardware/Software Interface" [17].

Patt and Patel's book is a bottom up approach to learning about computer architecture. The textbook starts by describing the digital components that make up the system, and then continues with more and more complex topics in computer architecture until the student begins to understand how a high-level program, written in the C language [18], is mapped to a processor and executed. Their approach is accompanied with the LC-2 computer simulator and tools that allows students to develop assembly programs and execute them on this machine. The instruction set architecture (ISA) is created for the LC-2 machine with a focus on providing the learner with a comprehensible language, but this ISA is not used in commercial processors.

Patterson and Hennessy's book approaches the same topic by providing a similar bottom approach that can be customized for students depending on if they are hardware or software focused. Their book is accompanied with software for the SPIM simulator [19] which simulates the MIPS [20] ISA. Patterson and Hennessy are also popularly known for a quantitative approach to learning computer architecture in their book, ''Computer architecture: a quantitative approach" [21], which is more focused towards students who will actually be working in the computer architecture field.

The prerequisites for the computer architecture course include introduction to programming and data structures, meaning all the students have been exposed to programming languages (normally JAVA programming). Students taking the lecture version of this course come from both computer science and computer engineering. For the badge-based course, there is an additional prerequisite in digital system design, which only computer engineers will have. This allows a the badge-based course to have less focus on number conversions, binary arithmetic, and digital circuits used in the computer architecture.

Some of the outcomes of our computer architecture course are listed here specifying that the student will have the ability to:

1) describe the operations performed by the CPU.
2) enumerate the registers in a CPU and describe their uses.
3) convert unsigned integers between the following representations: decimal, binary, octal, and hexadecimal.
4) represent signed integers using one's and two's complement representations.
5) perform addition and subtraction of signed integers represented in two's complement representation.
6) describe the salient aspects of the stored program concept.
7) describe the key components of a CPU and their functionality.
8) describe commonly used instructions, their formats, operands required, and encoding to opcodes.
9) illustrate the relationship between mnemonics and machine language translation.
10) use selected assembler directives for assembly language programming.

11) describe the various memory addressing modes used by the instructions with example usage.
12) describe the concept of CPI and quantitatively compare performance of various architectural solutions.
13) explain the concept and use of microprogramming
14) describe the process of linking and loading programs.
15) describe the various phases of assembly.
16) develop assembly language programs, debug assembly programs, and trace the operation of assembly language programs.
17) describe the process of interrupt handling.
18) describe the concepts and relationships between physical and virtual memory.
19) describe the four stages in a traditional pipeline
20) describe the taxonomy and categorization of computers into SISD, SIMD, MISD, and MIMD architectures.
21) describe the concept of superscalar processors
22) describe conventional bus and network connections.

From a more general perspective, we might state that once completing this course, students will be able to build simple assembly programs and understand how these programs run on a processor so that they may progress to understanding and performing in courses on Operating Systems, High Performance Computing, and Compilers.

3.2 Lecture Course Organization

Table 1 shows the evaluation structure of the lecture based course. This is a traditional course layout where quizzes and assembly programming assignments are spread out throughout the semester, and exams are used to assess the progress of students. The assembly assignments increase in complexity ending with a task of converting a ’C’ program into assembly instructions that includes string manipulation, dynamic memory allocation, and function calls. The assembly language used is MIPS, and a basic architecture is discussed that consists of the four phases of: instruction fetch, instruction decode, execute, and write back. This model can be understood by both the computer scientist and the computer engineer, but some of the finer details such as multiplexing and binary addition are briefly studied to allow the computer scientists to have some understanding of how the computer operates without taking a full course on digital system design.

One observation of this course organization, regardless of right or wrong, is that more than half the grade is allocated to test based assessment (60%). The other half of the course evaluation is split between programming and in class activities (some based on our previous work [22]). Because of this structure, a lot of time is spent in administering and preparing for tests.

3.3 Badge-based Course Organization

The badge-based course consists of a student demonstrating their increasing understanding of computer architecture by completing badges. The structure of these badges is shown in Figure 1 where three core or basic badges are at the top of the diagram, intermediate badges are under them, and advanced badges are along the base. In this figure, solid lines connecting badges are prerequisites for higher level badges, and the dotted lines are used for related badges (as in you should be familiar with one of the dotted lines before pursuing the targeted badge).

Each of the badge details is maintained on a course WIKI, and Figure 2 shows a sample badge for the “assembler basic badge”. Note that there is a circular badge with a red border in the upper left corner, and this is the digital badge that we created and is displayed on a students wiki page to show what they have achieved. Once a student has prepared the badge, they will bring the badge deliverables to the instructor and a discussion will occur to verify that the material is sufficiently mastered and understood. If the student has mastered the badge material, then the badge will be added to the students set of completed badges such as in Figure 3, which is an example of a students badges at the completion of the course in 2013 (3 core badges, Intermediate System Skills, Intermediate Assembly, and Advanced Compiler).

Class time in a course such as this, is spent working on badges with the added bonus that the instructor is available to answer questions. The instructor also evaluates each badge during class time or during office hours. The remaining logistical question is how are the badges related to course grades. To achieve this we use a mapping from badges to grades that is provided at the beginning of the semester in the syllabus. Table 2 shows how the badges are mapped to letter grades, and this is done for administration and economical purposes of the university. In terms of an educational benefit, the authors do not believe grading has any benefit in the learning process.

Table 1

LECTURE COURSE EVALUATION AND STRUCTURE

Item	Percentage of Course Grade	Additional Details and Topics Covered
Exam I	15%	1-11
Exam II	15%	1-18
Exam III	20%	All Topics
Class Participation	15%	Includes attendance and in-class execution of programs [22]
Quizzes	10%	Each quiz worth 1%
Assignment I	6%	MIPS Assembler programs
Assignment II	6%	built
Assignment III	6%	on
Assignment IV	7%	MARs Emulator

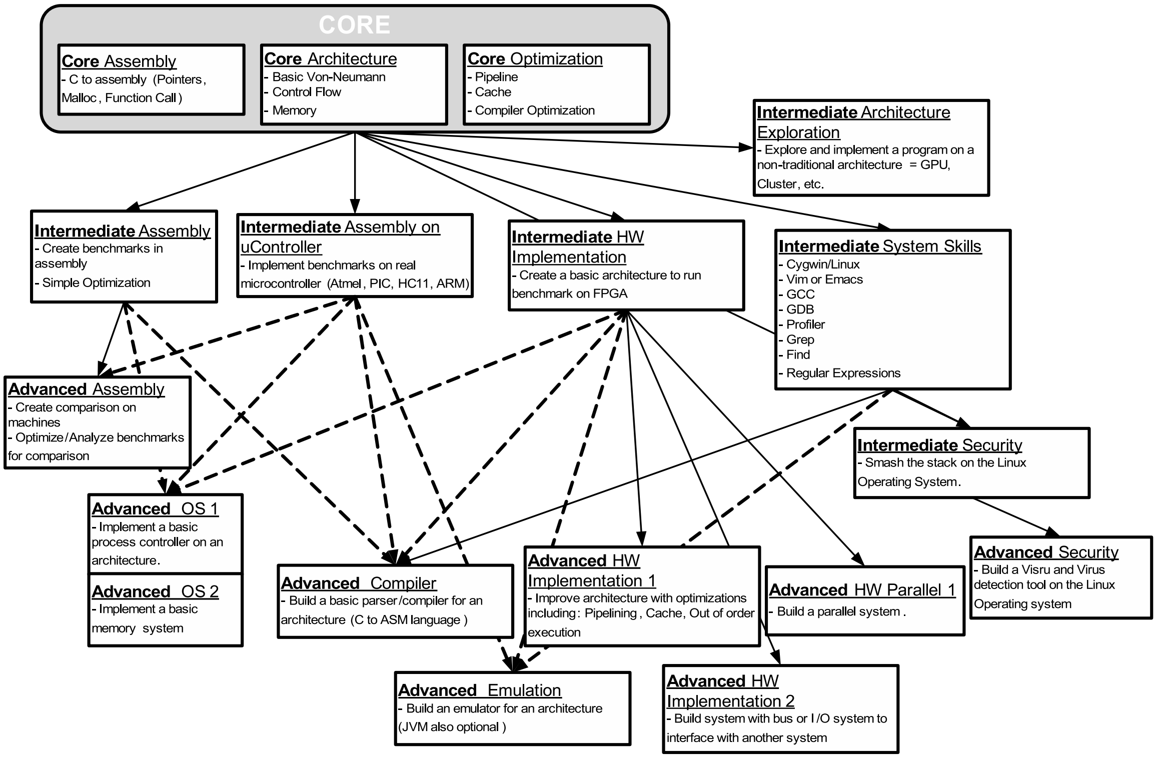

CORE
Core Assembly
- C to assembly (Pointers, Malloc, Function Call)
Core Architecture
- Basic Von-Neumann
- Control Flow
- Memory
Core Optimization
- Pipeline
- Cache
- Compiler Optimization
Intermediate Architecture Exploration
- Explore and implement a program on a non-traditional architecture = GPU, Cluster, etc.
Intermediate Assembly
- Create benchmarks in assembly
- Simple Optimization
Intermediate Assembly on uController
- Implement benchmarks on real microcontroller (Atmel, PIC, HC11, ARM)
Intermediate HW Implementation
- Create a basic architecture to run benchmark on FPGA
Intermediate System Skills
- Cygwin/Linux
- Vim or Emacs
- GCC
- GDB
- Profiler
- Grep
- Find
- Regular Expressions
Advanced Assembly
- Create comparison on machines
- Optimize/Analyze benchmarks for comparison
Intermediate Security
- Smash the stack on the Linux Operating System.
Advanced OS 1
- Implement a basic process controller on an architecture.
Advanced OS 2
- Implement a basic memory system
Advanced Compiler
- Build a basic parser/compiler for an architecture (C to ASM language)
Advanced HW Implementation 1
- Improve architecture with optimizations including: Pipelining, Cache, Out of order execution
Advanced HW Parallel 1
- Build a parallel system.
Advanced Security
- Build a Visru and Virus detection tool on the Linux Operating system
Advanced Emulation
- Build an emulator for an architecture (JVM also optional)
Advanced HW Implementation 2
- Build system with bus or I/O system to interface with another system

Fig. 1

THE PRE AND CO-REQUISITES FOR THE BADGES IN COMPUTER ORGANIZATION.

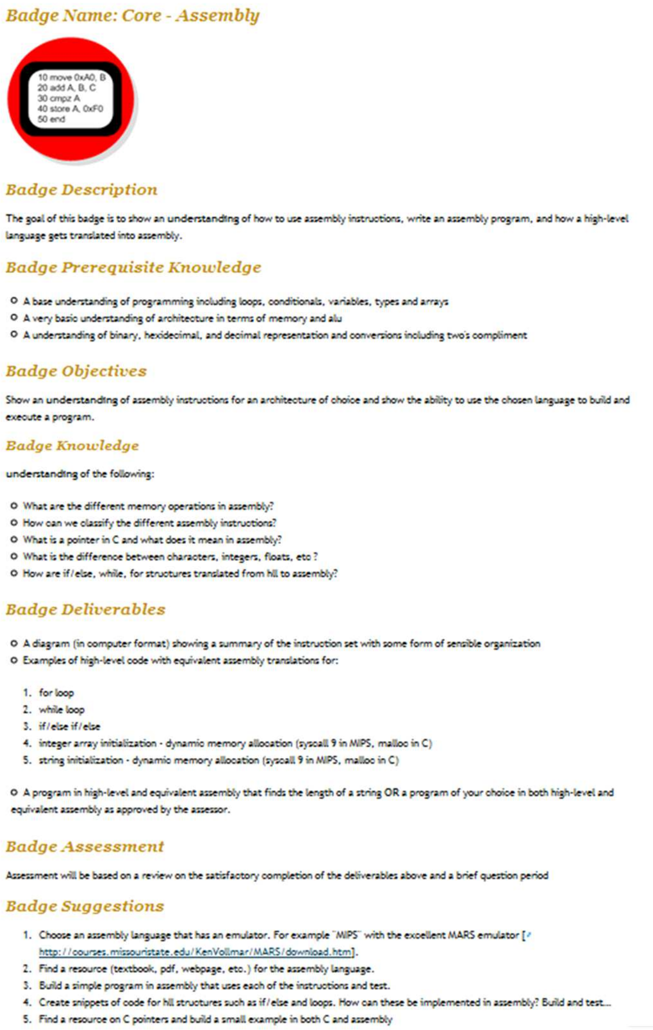
4. Results

The question we had for this course is not if the students would learn more or less in a badge-based course format (and we believe that students learn significantly more based alone on the topics and deliverables they produced). Instead, we were concerned that a mastery driven second year course would be biased all towards failure or all towards success as shown in the grade distribution. To observe this we compare the grades in the courses. Note that we currently do not have grades

Table 2

A TABLE SHOWING THE MAPPING BETWEEN BADGES AND LETTER GRADES.

Letter Grade	Core Badges	Intermediate Badges	Advanced Badges	Additional Details
F	0	-	-	-
D-	0-3	-	-	Base badges partially assessed
D	1	-	-	-
C	3	-	-	-
B	3	1	-	-
B+	3	2	-	-
A-	3	3	-	-
A	3	1	1	-
A+	3	1	1	Exceptional advanced work



Badge Name: Core - Assembly

Badge Description

The goal of this badge is to show an understanding of how to use assembly instructions, write an assembly program, and how a high-level language gets translated into assembly.

Badge Prerequisite Knowledge

- A base understanding of programming including loops, conditionals, variables, types and arrays
- A very basic understanding of architecture in terms of memory and alu
- A understanding of binary, hexidecimal, and decimal representation and conversions including two's compliment

Badge Objectives

Show an understanding of assembly instructions for an architecture of choice and show the ability to use the chosen language to build and execute a program.

Badge Knowledge

understanding of the following:

- What are the different memory operations in assembly?
- How can we classify the different assembly instructions?
- What is a pointer in C and what does it mean in assembly?
- What is the difference between characters, integers, floats, etc ?
- How are if/else, while, for structures translated from hll to assembly?

Badge Deliverables

- A diagram (in computer format) showing a summary of the instruction set with some form of sensible organization
- Examples of high-level code with equivalent assembly translations for:

  1. for loop
  2. while loop
  3. if/else if/else
  4. integer array initialization - dynamic memory allocation (syscall 9 in MIPS, malloc in C)
  5. string initialization - dynamic memory allocation (syscall 9 in MIPS, malloc in C)

- A program in high-level and equivalent assembly that finds the length of a string OR a program of your choice in both high-level and equivalent assembly as approved by the assessor.

Badge Assessment

Assessment will be based on a review on the satisfactory completion of the deliverables above and a brief question period

Badge Suggestions

1. Choose an assembly language that has an emulator. For example "MIPS" with the excellent MARS emulator [http://courses.missouristate.edu/KenVollmar/MARS/download.htm].
2. Find a resource (textbook, pdf, webpage, etc.) for the assembly language.
3. Build a simple program in assembly that uses each of the instructions and test.
4. Create snippets of code for hll structures such as if/else and loops. How can these be implemented in assembly? Build and test...
5. Find a resource on C pointers and build a small example in both C and assembly

Fig. 2

SAMPLE SCREEN SHOT OF ONE OF THE BADGES.

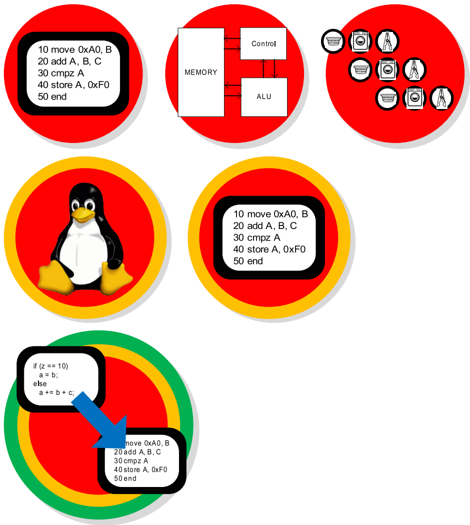


Fig. 3

A SAMPLE OF BADGES COMPLETED BY ONE STUDENT IN 2013.

for the 2014 class, but these will be included in our final camera ready version.

Figure 4 show the course grade distribution for each of the years the course was taught (2013 and 2014 the course was taught in the badge-based form). The sample size is significantly different when comparing 2010 and 2011 to 2013 and 2014 because the badge-based course was taught to computer engineers only and not the combination of computer engineers and computer scientists. This difference in students could impact the quality of there grades, but this factor was not studied. Also, no significant conclusions should be made from this data as the sample size is very small for the 2013 and 2014 classes. Finally, these results are for classes

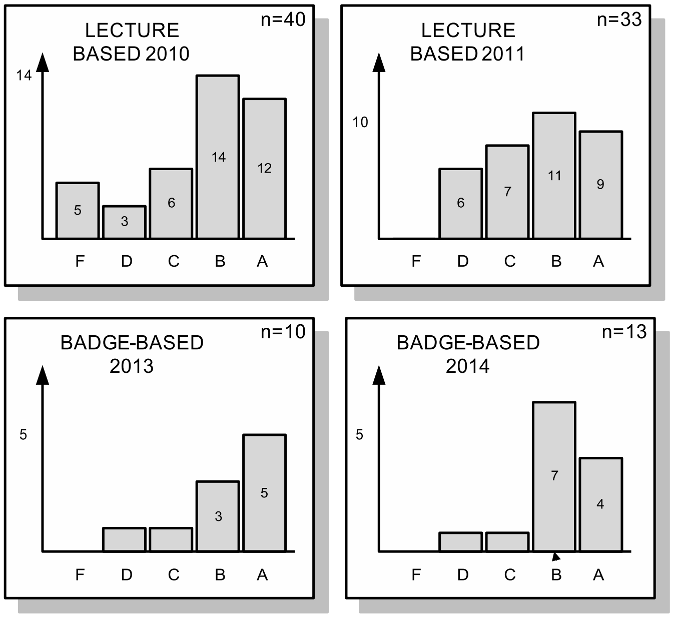


Fig. 4
THE FOUR GRADE DISTRIBUTIONS FOR 2 LECTURE-BASED AND 2 BADGE BASED RESULTS OF THE COURSE.

taught by the same instructor.

The most significant difference between the grades for each of the classes is that there are more A grades compared to B grades for the badge-based course when compared to the lecture-based course. We explain this to the fact that grades in the badge-based course are a long term decision made by the student as opposed to smaller decisions that might result in poor performance on a test or assignment. This is one of the key benefits to a mastery based approach since the student is under no pressure (other than the time in the semester) to perform at very specific assessment times.

An interesting point in terms of a student's choice is that a small percentage of the 2013 and 2014 class selected to just learn the minimal amount of material to get a passing grade. In the badge-based course, this choice had no impact on the other students as they could continue at their own pace on the material that they had chosen to explore deeper. Otherwise, there are not significant differences between the grade distributions. This suggests that a badge-based approach when mapped to grades will show a similar grade distribution when compared to a lecture based course.

5. Discussion

Our results show that the badge-based course has little impact on grade distributions, which from our perspective is good. If our results were biased towards all high letter grades, we, similarly, would be fine as we look to grades as an indication of mastery of the material and not as a categorizing mechanic for which students are relatively rated against one another. The exciting part of this method is the actual student experience, which is independent of much of the administration and organizational data that are described in this paper.

We start this discussion by comparing the core badges to material covered in the lecture-based course. The main difference between these materials is not the content, but how much of the course is spent learning these materials; in the badge-based course these materials are normally completed by the sixth week of study while in the lecture-based course this takes almost the entire semester. Note that the assessment choice, exams and quizzes, take up almost one quarter of the lecture based course in doing and preparing for the exams. We are not claiming that this time is wasted, but in the badge-based course this time is used by the student to learn other topics and to delve into an area of the material that they are most interested in. When considering that these students are training to become professionally practicing engineers explaining deliverables is much more career related than taking one-off tests.

One question the reader might asks is, how does the depth of the material covered in the core badges compare to the entire lecture course? We do not have any conclusive evidence to answer this question, but anectodally, we believe that the badge-based student learners gain almost the same understanding with less assembler programming skill that is regularly practiced in the lecture-based course. However, this low-level programming practice has little industrial impact, in most student's future careers, and spending that design practice on implementing hardware in register transfer level (RTL), assembly programming on microprocessors, learning about architecture and security, or even building and comparing assembly programs are all valuable activities that will help a student better understand computer architecture.

The additional difference with the badge-based course is the advanced badges. These badges allow a student to make significant strides into building and understanding complex pieces of a modern computer. This can be similarly achieved in a lecture based course with a summative project, but the logistic challenges to allow for the wide range of advanced badges to be tackled by all the students causes challenges with this approach. Many computer architecture and organization course with final projects normally focus on advanced assembly programming assignments. The badge-based approach allows students to select the piece of the system that they think they are most interested in.

One final discussion point is the scalability of the badge-based approach. From our perspective, the goal is not scalability, but the concept is still interesting to address and is of interest to the economics of engineering education. The badge-based course currently allows for the instructor to divide the time in the class adequately with all students such that for almost half of the class time there is no student demand. Based, on these rough numbers, we believe that we could hold approximately 25 students without any major reorganization steps. We do not believe that a class size of much greater than 25 can accommodate the students sufficiently without additional teaching assistants or instructors, however, this course could be grown with teaching assistants.

6. Conclusions

In this study, we investigated if there would be significant grade differences between a badge-based learning course and a traditional lecture-based course on computer architecture. I, preliminary, findings show that the grade distributions are very similar noting that the badge-based course was taught only to computer engineers (not computer scientists) and the sample sizes of the two approaches are significantly different. We, also, provided a framework for badge-based courses and discussed what benefits this approach has for students and instructors.

We believe that the badge-based approach to teaching computer architecture has made a significant improvement to what the students are learning. This is the case since badge-based learning allows interested learners to more deeply pursue aspects of computer architecture that they are interested in, which is not really possible in the lecture based approach.

References

[1] J. Macias-Guarasa, J. Montero, R. San-Segundo, A. Araujo, and O. Nieto-Taladriz, “A project-based learning approach to design electronic systems curricula,” *Education, IEEE Transactions on*, vol. 49, no. 3, pp. 389 –397, 2006. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1668283

[2] D. A. Kolb *et al.*, *Experiential learning: Experience as the source of learning and development*. Prentice-Hall Englewood Cliffs, NJ, 1984, vol. 1.

[3] R. J. Mislevy and M. Riconscente, “Evidence-centered assessment design,” *Handbook of test development*, pp. 61–90, 2006.

[4] M. Heritage, “What do teachers need to know and do,” *Phi Delta Kappan*, vol. 89, no. 2, pp. 140–145, 2007.

[5] Mozilla, “https://wiki.mozilla.org/Badges/About,” 2014.

[6] D. M. . L. Competition, “http://dmlcompetition.net/,” 2014.

[7] S. Abramovich, C. Schunn, and R. M. Higashi, “Are badges useful in education?: it depends upon the type of badge and expertise of learner,” *Educational Technology Research and Development*, vol. 61, pp. 217–232, 2013.

[8] F. Khaddage, R. Baker, and G. Knezek, “If not now! when? a mobile badge reward system for k-12 teachers,” in *Society for Information Technology & Teacher Education International Conference*, vol. 2012, no. 1, 2012, pp. 2900–2905. [Online]. Available: http://www.editlib.org/p/40029

[9] J. L. Santos, S. Charleer, G. Parra, J. Klerkx, E. Duval, and K. Verbert, “Evaluating the use of open badges in an open learning environment,” in *Scaling up Learning for Sustained Impact*. Springer, 2013, pp. 314–327.

[10] A. Bruckman, E. Edwards, J. Elliott, and C. Jensen, “Uneven achievement in a constructionist learning environment,” in *International Conference of the Learning Sciences: Facing the Challenges of Complex Real-world Settings*, vol. 7, no. 17. Psychology Press, 2013, p. 157.

[11] S. Charleer, J. Klerkx, J. L. Santos, and E. Duval, “Improving awareness and reflection through collaborative, interactive visualizations of badges,” in *3rd Workshop on Awareness and Reflection in Technology-Enhanced Learning*, 2013. [Online]. Available: http://ceur-ws.org/Vol-1103/paper5.pdf

[12] J. Young, “Badges earned online pose challenges to traditional college diplomas,” *The Chronicle of Higher Education*, 2012.

[13] J. Antin and E. F. Churchill, “Badges in social media: A social psychological perspective,” in *CHI 2011 Gamification Workshop Proceedings (Vancouver, BC, Canada, 2011)*, 2011.

[14] D. Gibson, N. Ostashewski, K. Flintoff, S. Grant, and E. Knight, “Digital badges in education,” *Education and Information Technologies*, pp. 1–8, 2013.

[15] A. Kohn, *Punished by rewards: The trouble with gold stars, incentive plans, A’s, praise, and other bribes*. Houghton Mifflin Harcourt, 1999.

[16] Y. N. Patt and S. J. Patel, *Introduction to Computing Systems: From Bits & Gates to C & Beyond*. New York, NY, USA: McGraw-Hill, Inc., 2004.

[17] D. Patterson and J. Hennessy, *Computer Organization and Design: The Hardware/software Interface*. Morgan Kaufmann, 2005.

[18] B. W. Kernighan and D. M. Ritchie, *The C programming language*. Prentice-Hall, 1978.

[19] J. Larus, “SPIM S20: A MIPS R2000 Simulatora,” Tech. Rep., 1990.

[20] J. Hennessy, N. Jouppi, S. Przybylski, C. Rowen, T. Gross, F. Baskett, and J. Gill, “Mips: A microprocessor architecture,” in *MICRO 15: Proceedings of the 15th annual workshop on Microprogramming*. Piscataway, NJ, USA: IEEE Press, 1982, pp. 17–22.

[21] J. L. Hennessy and D. A. Patterson, *Computer architecture (2nd ed.): a quantitative approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.

[22] P. Jamieson, D. Davis, and B. Spangler, “The mythical creature approach-a simulation alternative to building computer architectures.” in *FECS*, 2010, pp. 23–28.