A Survey of Assignments in Undergraduate Computer Architecture Courses

http://dx.doi.org/10.3991/ijet.v11i06.5776

Dimitris Kehagias
Technological Educational Institute (T.E.I.) of Athens, Athens, Greece

***Abstract*—Computer architecture is an essential topic in undergraduate Computer Science (CS) curricula. Teaching computer architecture courses to CS students can be challenging, as the concepts are on a high abstraction level and not easy to grasp for students. Learning of computer architecture abstracts is strongly reinforced by hands-on assignment experience. This paper presents results from a survey of assignments from 40 undergraduate computer architecture courses, which are offered in 40 CS departments. These surveyed courses are selected from universities listed among the 120 top North America universities by the Webometrics Ranking of World Universities 2015. The information used for this survey is based solely on material publicly accessible on the websites of courses.**

***Index Terms*—Assignments, Computer Architecture Courses, Computer Science, Survey.**

I. INTRODUCTION

Understanding the interaction of hardware with software should be considered a minimum in knowledge for any CS graduate. They should not conceive computer as a black box that executes programs in a magic, automatic way. To this end, computer architecture and organization is an important area in undergraduate CS curricula. This area is acknowledged as a substantial part of the body of knowledge in computer science by IEEE Computer Society and Association for Computing Machinery in their Joint Task Force on Computing Curricula [1]. According to this proposal the knowledge area of "architecture and organization" (AR) contains core and elective topics. The core topics are: (a) digital logic and digital systems, (b) machine level representation of data, (c) assembly level machine organization, (d) memory system organization and architecture, and (e) interfacing and communication, while the elective topics are: (f) functional organization, (g) multiprocessing and alternative architectures, and (h) performance enhancements. The same proposal states that "*the core topics are intended to support programs that elect to require only the minimum of computer architecture of their students. For programs that want to teach more than the minimum, the same AR topics can be treated at a more advanced level by implementing a two-course sequence. For programs that want to cover the elective topics, those topics can be introduced within a two course sequence and/or be treated in a more comprehensive way in a third course*" [1].

Regardless of the number of course sequence in computer architecture offered in an undergraduate CS program, the individual topics taught involve many abstract concepts for a student and as research highlights in [2] the difficulties encountered by students studying computer architecture have been well recorded. Teaching computer architecture courses to CS students can be inefficient if the teaching methods focus only on the theoretical aspects [3, 4]. A major problem in teaching the courses is how to help students make the reasoning that connects their theoretical knowledge with practical experience [5, 6].

Learning of computer architecture abstracts is strongly reinforced by hands-on assignment experience. In the context of a computer architecture course, students gain an in-depth understanding of the inner design and operation of a modern microprocessor and trade-offs that are present at the hardware/software interface, by applying classroom knowledge in a series of assignments. These assignments can be a combination of homework (paper and pencil problem sets), and/or labs, and/or projects.

It is usually a challenge to teach a subject in an environment where the covered topics are advancing very rapidly. In such conditions, the instructors must be up to date with the state of the art. At the same time, they should continuously revise their lectures, problem sets, and lab-project assignments to match the new development in the covered topics. For example, in the new multicore and many-core era many institutes and universities change their curriculums of computer architecture courses and hence the contents of the provided assignments. Should be noted also that the level and the specific contents of topics taught in computer architecture courses vary from institution to institution. Therefore, many approaches may exist in the type and content of the assignments given in a computer architecture course.

In the context of an undergraduate computer architecture course introducing appropriate hands-on lab and project assignments besides the lectures will make the course more interesting and will provoke the students to dive more deeply to learn that no magic is required to make a computer work. Liang [7] performed a nice survey of hands-on assignments and projects. Concerning practical contents of hands-on assignments several simulators are usually used. A good survey on simulators suitable for teaching courses in computer architecture is presented by Nikolic et al. [8].

In this paper, we present an overall and up-to date picture of various types of assignments in undergraduate computer architecture courses, their categorization, as well as programming languages, tools and platforms used in these assignments. To this end, the author surveyed assignments collected from 40 undergraduate computer architecture courses. These surveyed courses are selected from 40 CS departments of universities listed among the 120 top North America universities by the Webometrics Ranking of World Universities 2015 [9]. The information used for this survey is based solely on material publicly accessible on the websites of courses.

The rest of the paper is organized as follows. In section II we categorize the assignments based on their contents and tools used. Section III presents the results for the assignments surveyed. Section IV concludes the paper.

II. ASSIGNMENT CATEGORIZATION

Assignments included in computer architecture courses can be categorized based on their contents and tools used. With this context, the surveyed assignments fall within four major categories: (a) problem sets, (b) assembly language programming (c) computer architecture design & test using HDL-based environments and (d) exploring computer architecture topics using high level programming & instrumentation tools. The last two major categories are divided into several subcategories as shown in Table I. There are totally eight subcategories.

TABLE I.
ASSIGNMENT CATEGORIZATION

Main categories	Subcategories
I. Problem sets	Pencil and paper problems
II. Assembly language programming	Coding in assembly language
III. Computer architecture design & test using HDL-based environments	1. Basic digital logic design and test
	2. Single-cycle processor
	3. Basic pipelined processor
	4. Cache hierarchy
IV. Exploring computer architecture topics using high level programming & instrumentation tools	5. Pipelining
	6. Caches
	7. Branch predictors
	8. Set-up/modify simulators

A. *Problem sets*

This category includes pencil and paper problems. Homework is assigned throughout a semester and typically consists of problems that come from the textbook, with other supplementary problems added. Rarely this type of problems is given as preparation for the lab assignments.

B. *Assembly language programming*

The ISA (Instruction Set Architecture) abstraction layer in computer systems is the critical interface between software and hardware. Specific ISAs are used in any computer architecture course to help students in learning the fundamentals of computer architecture. Assembly language programming is the best way to study an ISA. Assignments that involve programming in assembly language using a simulator are included in this category. Usually the assembly language programming includes: mapping of high-level language constructs into assembly code, addressing modes and their relation to arrays, integer and floating point arithmetic, subprograms, parameters, linkage to high level languages, interaction between assembly language programs and system calls, interrupts and I/O, and the assembly process.

C. *Computer architecture design & test using HDL-based environments*

In the third category fall assignments whose implementation requires appropriate tools and platforms to enable the students first to design specific computer system configurations and then simulate and -in some cases- implement them. The majority of tools and platforms used are HDL-based. HDLs (hardware description languages) are programming languages used to describe a circuit behaviorally, structurally or both. Students use a HDL and an associated simulator to design components of computer systems and explore architectural concepts.

The first subcategory deals with basic digital logic design. Although logic design usually taught in separate standalone courses in CS curricula, many computer architecture courses include the basics of logic design, starting with simple combinational circuits and building up to state machines. The knowledge from this subcategory is the basic foundation for students to understand better computer architecture.

The second subcategory deals with building the datapath and control of a single-cycle processor sufficient to implement an instruction set like MIPS, and with verifying its correctness. In the third and fourth subcategories appropriate environments are used in order for students to experiment and understand the principles and complications of pipelines and caches respectively.

D. *Exploring computer architecture topics using high level programming & instrumentation tools*

The fourth category contains assignments that address specific topics, not necessary different from the topics that the third category addresses, whose implementation requires appropriate tools and frameworks that enable the students to simulate, instrument and modify already created systems as well as to setup various simulators and evaluate their behavior. Students are provided various implementations where they experiment on architectural techniques such as branch predictors, caches, and pipelining, by writing their code in a high level language.

III. SURVEY RESULTS

The surveyed assignments were collected from 40 undergraduate computer architecture courses from 40 CS departments of universities listed among the 120 top North America universities by the Webometrics Ranking of World Universities 2015 [9]. Twenty one of the surveyed courses were taught during the fall 2015 semester, seventeen during the spring 2015 period and two during the spring and fall 2014 semesters. Prerequisite for a course to be surveyed was to have a detailed information web page publicly accessible. A course was selected on the basis of its description and whether or not the contents of included assignments were available. For each selected course we examined the contents of all assignments and we placed it under the appropriate four categories and eight subcategories, as shown in Table II. In addition, we recorded programming languages, tools and platforms used in these assignments as well as books suggested by instructors.

Thirty two courses have a main textbook that is required together with other reference material, while 7 courses have an optional textbook. The textbook "Computer Architecture: A Quantitative Approach" by John Hennessy and David Patterson [10] was required by 7 courses while the textbook “Computer Organization and Design: The Hardware/Software Interface” also by David Patterson and John Hennessy [11] was required by 23 courses.

A. Assignment distribution

For each one of the forty surveyed courses, Table II shows the assignment distribution under the four categories and the eight subcategories presented in section II. The column WT shows the weight percentage that assignments contribute to the final grade of students. For the course 03 the weight is not given. As an example take from Table II the course 05. It contributes 50% of the final grade and has assignments in categories I, II, III and IV. In the last two categories it has assignments in subcategories 1, 5, 7 and 8.

TABLE II.
ASSIGNMENT DISTRIBUTION

A/A	**WT (%)**	**Assignment main categories & subcategories**									
		I.	***II.***	***III.***				***IV.***			
				1	2	3	4	5	6	7	8
01	25								×	×	×
02	25	×						×	×	×	×
03		×						×	×	×	×
04	45	×		×	×	×	×				
05	50	×	×	×				×		×	×
06	50	×	×	×	×						
07	55								×	×	×
08	50			×	×	×	×				
09	30	×		×	×	×	×				
10	45	×	×								×
11	45	×	×	×	×						
12	45	×									
13	50	×	×	×	×						
14	40	×	×	×	×	×					
15	45	×		×	×	×	×				
16	50	×									
17	40	×						×	×	×	×
18	100	×									×
19	10	×							×		
20	40	×		×	×	×					
21	18		×								
22	40	×						×			
23	70	×	×						×	×	
24	25	×									
25	55	×			×						×
26	20	×	×								
27	20	×									
28	25	×	×	×							
29	60	×	×	×							
30	20	×	×								
31	30	×	×								
32	30	×						×	×		×
33	15	×	×								
34	55	×	×								
35	40	×									
36	30	×	×								
37	30	×						×			
38	25	×	×								
39	20	×									
40	40	×	×								

According to the assignment distribution for each course as shown in Table II, the distribution of courses on the 4 categories and the 8 subcategories are shown in Figs 1 and 2 respectively. As shown in Fig. 1, fourteen courses have assignments in category IV, thirteen in category III, eighteen in category II and thirty six in category I. Assignments in the problem sets (category I) and the assembly language programming (category II) are most popular. Taking an example in Fig. 2 it can be seen that six courses have assignments in subcategory three. Subcategories 1, 2 and 8 are most popular.

According to the assignment categorization for each course as shown in Table II, the number of courses that cover different numbers of subcategories and categories are shown in Figs 3 and 4 respectively. From Fig. 3 it can be noticed that eight courses have assignments in four different subcategories, while there is no course having assignments in more than four different subcategories. As shown in Fig. 4 only one course has assignments in all categories, while twenty courses have assignments in two different categories.

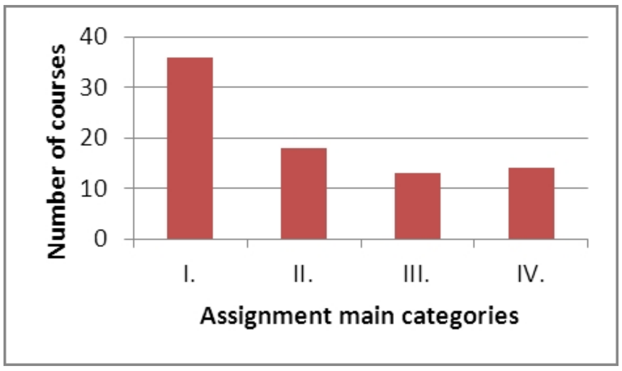

Figure 1. Course distribution over the 4 categories

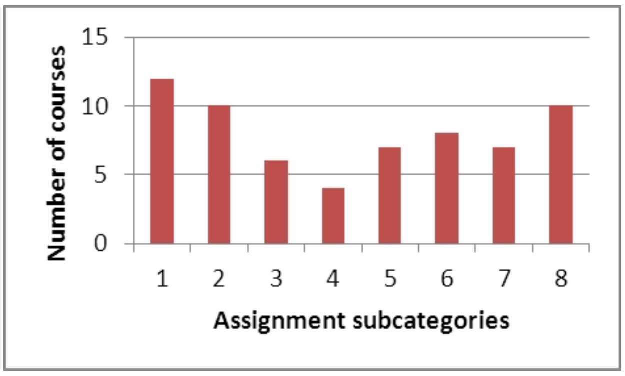

Figure 2. Course distribution over the 8 subcategories

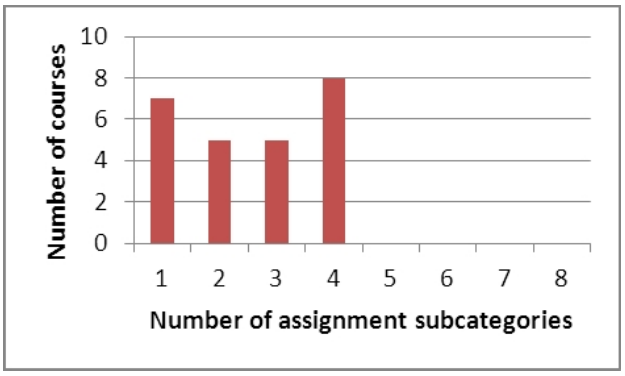

Figure 3. Course distribution over subcategory coverage

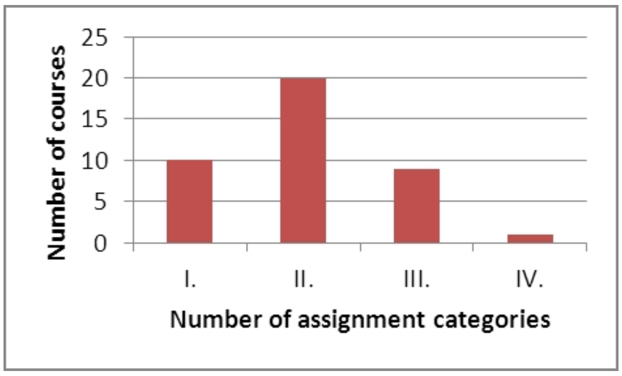

Figure 4. Course distribution over category coverage

B. *Problem sets*

Thirty six of the surveyed courses have assignments with problem sets and 30 of them have additional assignments in other categories or subcategories.

C. *Assembly language programming*

The assembly programming is centered on the MIPS processor (ISA) [11], which is a well-known processor (ISA) in the computer architecture academic community. Besides MIPS processor there are other processors that are targeted to the assembly programming such as LC-2K (Little Computer 2000) and Y86.

The LC-2K is a small example computer and Instruction Set Architecture (an 8-register, 32-bit computer with 65536 words of memory) used at the university of Michigan. Y86 is more RISK-like ISA based on x86 (IA-32) with simpler instruction formats and addressing modes.

The functional simulators used for interpreting, executing, and debugging assembly programs are SPIM [12], QtSPIM [12], and MARS [13]. SPIM is a self-contained simulator that runs MIPS32 programs. QtSPIM, the newest version of SPIM, is an open-source simulator, written in C++ and Qt that runs MIPS32 programs. MARS is an Integrated Development Environment (IDE) that simulates the execution of MIPS assembly programs.

D. *Computer architecture design & test using HDL-based environments*

Verilog [14] HDL is used by the vast majority of the surveyed courses to design and verify major components of a computer architecture, while four courses use a graphic way (Logisim [15]) for design and verification. Twelve of the surveyed courses have their first assignment dealing with the design of basic combinational circuits and classic memory elements. There is one surveyed course that uses Vivado FPGA design tool from Xilinx [16] in its assignments. Vivado is a commercial software suite for synthesis and analysis of HDL designs.

There are many HDL design and simulation tools used in the surveyed courses. Icarus Verilog (iverilog) is an open source Verilog simulation and synthesis tool for compiling Verilog models into simulators [17]. It can be used with open source GTKWave tool to simulate and view waveforms [18]. Instead of using iverilog-GTKWave tools, Modelsim [19] can be used as a multi-language HDL simulation environment. Logisim is an educational tool for designing and simulating digital logic circuits [15]. Synopsys VCS (Verilog Compiler Simulator) is a Verilog compilation tool and VirSim a graphical user interface to VCS for debugging and viewing waveforms [20].

The processors selected to design and verify their correctness in the surveyed courses using HDL-based environments are RISC-style processors such as MIPS. They may also be educational processors that come with a course. Such examples are processors for the PARCv2 ISA, the LC4 processor, PAW and the W450 processor. The PARCv2 instruction set architecture [21] is a subset of MIPS32, developed primarily for educational purposes. LC4 processor implements every instruction on the LC4 ISA designed and used at the University of Pennsylvania. At Princeton University students design, implement, and test a microprocessor which executes the PAW instruction set. W450 processor has an ISA somewhere between CISC and RISC used at the University of Waterloo.

E. *Exploring computer architecture topics using high level programming & instrumentation tools*

Ten of the surveyed courses include assignments consisting of computer architecture foundations such as pipelining, memory hierarchies and branch prediction that are worked out by high level coding using C, C++, Java, or Python and frameworks for the instrumentation of programs.

Students write simulators that are designed to display the functionality of pipelined processors along with performance statistics, or simulators for several cache architectures evaluating their performance. They also implement programs to simulate various branch predictors on a number of branch traces from real benchmark programs. The simulation infrastructure is built using a binary instrumentation tool called Pin [22]. It supports binary instrumentation of executables on all Intel platforms. Pin contains various tools for use, a few of which are the data cache, branch predictor simulators, and a tool to measure instruction counts and to analyze the latency of load instructions.

Other educational simulation tools used in the surveyed courses include Chisel and SSIM. Chisel is a simulation environment from UC Berkeley [23]. It implements an entire functioning processor (BOOM) where students run experiments on it and analyze the design. SSIM is a simulator that models the SEQ processor design, presented in the textbook “Computer Systems: A Programmer's Perspective”, by Randal E. Bryant and David O'Hallaron.

Under this category the processors selected to implement in the surveyed courses are processors similar to MIPS and x86 processors. They may also be an educational processor that comes with a course. Such examples are the processors RISC-V Berkeley BOOM and LC-2K. RISC-V Berkeley out-of-order Machine (BOOM), used in UC Berkeley’s computer architecture and engineering course, is heavily inspired by the MIPS R10k and the Alpha 21264 out-of-order processors [24, 25].

IV. CONCLUSIONS

Computer architecture courses in undergraduate CS curricula are usually accompanied by assignments so that students can better obtain a practical understanding of the topics lectured. This paper presents an overall and up-to date picture of various types of assignments used in undergraduate computer architecture courses at the top North

America universities. This survey might help educators to select and/or create assignments and tools for their computer architecture courses.

REFERENCES

[1] IEEE/ACM, "The Joint Task Force on Computing Curricula of Association for Computing Machinery (ACM) and IEEE Computer Society", *Computer Science Curricula 2013 Final Report.*

[2] N. Thomas, F. Carroll, R. Kop, and S. Stocking, "iBook learning experience: the challenge of teaching computer architecture to first year university students", In *Proceedings: WORLDCOMP'12- The 2012 World Congress in Computer Science, Computer Engineering, and Applied Computing*, July 16-19, 2012, USA.

[3] A. Y. El-Din, and H. Krad, "Teaching computer architecture and organization using simulation and FPGAs", *International Journal of Information and Education Technology*, Vol.1, No.3, 2011.

[4] A. Siddiqul, M. Khan, and S. Akhtar, "Supply chain simulator: A scenario-based educational tool to enhance student learning" *Computers & Education*, Vol.51, No.1, 2008, pp. 252-261. http://dx.doi.org/10.1016/j.compedu.2007.05.008

[5] R. Cassells, "What The Data Says About Generation Y Australian Chief Executive", *Melbourne, Australia: Committee for Economic Development of Australia (CEDA)*, 2007.

[6] C. Yehezkel, W. Yurcik, M. Pearson, and D. Armstrong, "Three simulator tools for teaching computer architecture: EasyCPU, Little Man Computer, and RTLSim", *ACM Journal of Educational Resources in Computing*, Vol.1, No.4, 2001, pp. 60-80. http://dx.doi.org/10.1145/514144.514732

[7] X. Liang, "A survey of hands-on assignments and projects in undergraduate computer architecture courses", In *Advances in Computer and Information Sciences and Engineering*, T. Sobh, Ed. Springer Netherlands, 2008, pp. 566–570. http://dx.doi.org/10.1007/978-1-4020-8741-7_101

[8] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization", *IEEE Transactions on Education*, Vol.52 No.4, 2009, pp. 449–458. http://dx.doi.org/10.1109/TE.2008.930097

[9] Webometrics Ranking of World Universities: http://www.webometrics.info/en/Americas/North_America

[10] John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th edition, Morgan Kaufmann, 2012.

[11] David A. Patterson and John L. Hennessey, *Computer Organization & Design: The Hardware/Software Interface*, 4th or 5th edition, Morgan Kaufmann, 2014.

[12] J. Larus, *SPIM: A MIPS32 Simulator*, http://spimsimulator.sourceforge.net.

[13] D. K. Vollmar and D. P. Sanderson, *MARS: An Education-Oriented MIPS Assembly Language Simulator*, March 2006, http://courses.missouristate.edu/KenVollmar/mars/fp288-vollmar.pdf

[14] Verilog, http://www.verilog.com/

[15] http://www.cburch.com/logisim/index.html

[16] http://www.xilinx.com/products/design-tools/vivado.html

AUTHOR

Dimitris Kehagias is with the Informatics Department, Technological Educational Institute of Athens, Ag. Spiridona, 12210, Athens, Greece (dkehayas@teiath.gr).

Submitted 19 April 2016. Published as resubmitted by the author 23 May 2016.