European Journal of Engineering Education
Vol. 35, No. 1, March 2010, 91–98

Taylor & Francis
Taylor & Francis Group

Switching from computer to microcomputer architecture education

Dimosthenis E. Bolanakis[a]*, Konstantinos T. Kotsis[a] and Theodore Laopoulos[b]

[a]*Department of Primary Education, Physics Education Laboratory, University of Ioannina, 3rd floor, (cc. Konstantinos Kotsis), Ioannina 45110, Greece;* [b]*Department of Physics, Electronics and Computers Section, Aristotle University of Thessaloniki, 1st floor, Thessaloniki 54124, Greece*

(*Received 29 June 2009; final version received 19 August 2009*)

In the last decades, the technological and scientific evolution of the computing discipline has been widely affecting research in software engineering education, which nowadays advocates more enlightened and liberal ideas. This article reviews cross-disciplinary research on a computer architecture class in consideration of its switching to microcomputer architecture. The authors present their strategies towards a successful crossing of boundaries between engineering disciplines. This communication aims at providing a different aspect on professional courses that are, nowadays, addressed at the expense of traditional courses.

Keywords: computer architecture; microcomputers; software engineering education

1. Introduction

Emerging technologies in the discipline of computing constantly increase the need to broaden education within computing curricula. Nowadays, software engineering education (i.e. the area of education concerned with the application of theory) features two main issues: (1) the oddity of introducing fresh students to methods of programming using virtual environments (Margush 2006) and (2) the demand of preparing students for the eventuality of becoming programmers of general-purpose computers or embedded systems (Anguita and Fernandez-Badomero 2007). The former issue is a result of new trends in programming languages that are increasing in complexity and expanding a numerous set of application programmer interfaces. The latter is a result of recent advances in embedded control that is inundating industry and everyday life with home automations, car systems, toys, etc.

In the face of rapid technological change, the authors considered 6 years ago a switching of the traditional computer architecture class to microcomputer architecture (a choice that intended to meet current trends in software engineering education). Following extensive cross-disciplinary research, a good working tutoring system for theory (Bolanakis *et al.* 2008, 2009a, 2009b, 2009c,) and practice (Bolanakis *et al.* 2007a, 2007b, 2007c) is presently experienced at the Department of Communications, Informatics and Managements, Epirus Educational Institute of Technology,

*Corresponding author. Email: dbolanis@cc.uoi.gr

ISSN 0304-3797 print/ISSN 1469-5898 online

DOI: 10.1080/03043790903312162
http://www.informaworld.com

Arta, Greece. Regarding the earned benefits from this particular research study, the present communication provides a detailed description of strategies needed for reforming (changing) the traditional computer architecture course. The author anticipate providing a different aspect on professional courses that are, nowadays, addressed at the expense of traditional courses.

The remainder of this paper is organised as follows. Section 2 justifies the reason for switching the computer architecture class to microcomputer architecture. Section 3 provides the background information of our research. Section 4 presents our strategies on crossing boundaries between engineering disciplines by means of a generalised working framework and Section 5 concludes the article.

2. Microcomputer architecture justification

Microcomputer technology is regularly addressed to electronic engineers, while traditional microcomputer courses are regularly carried out using a simplified and low-cost microcontroller, that is, an 8-bit microcontroller unit (MCU). Identically, Montanez (2005) observes the importance of sparking students' interest in MCUs (and in various levels of engineering education) by introducing the subject in a very simple form. This choice stems from the need to avoid the possible disorientation arising from the use of complex MCU architectures.

In a review of literature, one can observe the popularity of migrating small MCUs in various engineering disciplines so as to serve a wide range of monitoring and control applications. Some of the numerous examples are in line with the education of chemical engineering (Lodge 2006), mechanical engineering (Culbreth 2001, Giurgiutiu *et al.* 2004), biological and agricultural engineering (Hamrita and McClendon 1997, Hamrita 2002), etc. It is generally accepted that the responsibility of engineering educational departments is to properly train the future managers of the society in all aspects of a computer's working and usage (Burney and Abdul Haq 1991). Due to the widespread use of MCUs in every field of engineering discipline, the authors considered (6 years ago) extending the prevailing MCU tutoring methods suitable for electronic engineers to software engineering students in order to support issues related to their profession.

Software engineering, as defined by Wikipedia, is the application of engineering to software. Hazzan and Karni (2006) describe software engineering as a multifaceted discipline that entails different aspects from different professionals in the field. Moreover, Mengel and Carter (1999) report that the activity of software engineers is multidisciplinary and therefore, students should learn to interact with professionals from many different backgrounds in order to succeed in building interdisciplinary systems. Given the technological evaluation of embedded computer systems in industry and everyday life, Anguita and Fernandez-Badomero (2007) suggest preparing software engineering students for the probability of becoming programmers of general-purpose computers or embedded computers.

Reasons given above motivate us to enquire into the manner in which a microcontroller-based tutoring system could be brought into harmony in the software engineering curriculum. A methodological practice on embedded control is proven to reinforce the educational level of the software engineering students on the hardware domain (Bolanakis *et al.* 2007c). Although students may not become Embedded Computer Systems (ECS) programmers, they gain significant experience in interacting with engineers specialising in hardware design issues. It is worth noting Sparks (1993) observations on the requirement for less specialisation in graduates (because of the rapidly changing technology). Harmonising microcomputer technology with computer architecture learning is a concern of the issues presented hereafter.

In the last decade, the spiralling growth of basic content areas in computing curricula has absorbed the assembly language programming class into the computer architecture class; an option that relies on the need to make room in the curriculum for emerging technologies (Agarwal

and Agarwal 2004). To date, assembly language programming is regularly carried out in the introductory computer architecture class using simulated platforms that are addressed to substitute the actual processors (Osborne 2002, Wolf *et al.* 2002). This alternative arises from the complexity of modern/sophisticated processors (such as a Pentium 4) and subsequently from their unsuitability in supporting an introductory assembly language programming course. However, many instructors oppose this direction and sustain the importance of retaining the assembly language course, as it consists of a catalytic course for understanding the programming mechanisms (Buckner 2006). Moreover, it is generally accepted that (compared with the real hardware) the simulator alone does not provide a real sense of accomplishment to the students (Hamblen *et al.* 1990).

There is abundant literature showing already in the 1980s the influential linkage between microcomputer technology and assembly level programming due to the limited instruction set (and simplified) architecture (Silver and Leeper 1983, Tran and Robillard 1985). This distinctive attribute of microcomputers (and embedded computers in general) renders assembly language programming applicable until today. Under these circumstances, many instructors have recently been directed towards the option of retaining an assembly language course, while incorporating the use of MCU in the tutoring process (Maurer 2005, Margush 2006). However, our lines of research challenged a more audacious attempt.

In order to make room in the curriculum for emerging technologies, our research focused on switching the traditional computer architecture class to microcomputer architecture. Above all, it is in our belief that the assembly language learning appropriately and effectively serves the needs of computer architecture learning and thus its absorbance into the computer architecture class consists of an ideal scheme. At this point it is worth noting Stallings' (2003) and Duntemann's (1992) definitions on (1) 'computer architecture' and (2) 'assembly language' terms. The former term refers to all the – 'visible to the programmer' – features that have an impact on the execution code. The latter refers to the total control of the assembly instruction mnemonics over every individual machine instruction. Consequently, assembly language programming addresses the programmer to a deeper insight into the processor's mechanisms.

Although retaining the traditional assembly language course and revising it into a MCU-based tutoring system could be considered straightforward the switching of computer architecture classes to microcomputer architecture requires precious endeavour. On the basis of Borrego and Newswander's (2008) remarks on interdisciplinary collaboration as integration of epistemologies, the former initiative could be characterised as a multidisciplinary approach, while the latter as a truly interdisciplinary approach (as it is enforced by researchers from different disciplines, combining their knowledge to work in a more integrated way towards a solution). It is worth noting that the accomplishment of the present (long term) cross-disciplinary research relied on the collaboration of instructors across disciplines of electronics, informatics, automation, didactics, and applied physics. In the rest of the article the authors share their experience in building bridges across disciplines and provide in detail, the necessary information for switching computer architecture classes to microcomputer architecture.

3. Background research

3.1. *The laboratory class*

Due to the need to reinforce software engineering education on embedded control, and subsequently upgrade students' educational level on the hardware domain, our research originated from the structural change of the laboratory training. In technology-related courses, it is important to assure students' experiential learning, while avoiding drawing students' attention away from the goal of the course by focusing on too much technical detail.

Under these conditions, the design of a technically accurate educational board system (EBS) as well as the development of an effective set of experiments were taken into consideration. Through an extensive study of hands-on experiments education, our strategies relied on the triptych of functions 'user-friendly, instructive, and flexible' for the design of proper and effective EBSs (Bolanakis *et al.* 2007a). Regarding this triptych, custom-made EBSs should be characterised by an explicit and expandable geometry (that allows a minimum amount of work on hardware implementation during the class) along with a potential number of practical examples that guarantee experiential learning.

In order to retrieve information derived from students' responses during the laboratory training, more than a year-long evaluation of the proposed EBS and set of experiments within the (1 day a week, for 2 h schedule) class followed. In that period, students' comprehension difficulties were mainly identified by the obscure linking between the firmware and the hardware, which was subsequently causing confusion during the code development process.

A promising answer for overcoming barriers of understanding was revealed in Levin's (1981) suggestions on picture functions and in particular on the interpretational picture function. The interpretational function is regularly addressed for the difficult-to-follow concepts and is proved to aid understanding. Following Levin's suggestions, Mayer and Gallini (1990) strongly recommend these kinds of pictures for the explanation of scientific issues. As a consequence of these suggestions, our textbooks and power point lectures were enhanced with parts-and-steps examples depicting the interaction between firmware and hardware. Accordingly, our strategies were concerned with extending the prevailing picture examples on microcontroller education (that are regularly referred to as schematic diagrams) to students with insufficient background on hardware design issues (Bolanakis *et al.* 2007b). Thus, the overall work on laboratory training relied on a microcontroller-based tutoring system for integrating computer architecture laboratory courses (Bolanakis *et al.* 2007c). Having overcome issues related to the laboratory class, a harmonic theory course was taken into consideration.

3.2. *The theory class*

Regarding the theoretical part of the course, our interest focused on issues related to the processor's instruction set architecture (ISA) such as op-codes, addressing modes, etc., which were purposely excluded from the laboratory class. Although the study of ISA is fundamental for an effective experiential learning, the onslaught of new information (in a cross-disciplinary approach) would probably address students to a state of confusion and disorientation from the main purpose of the course. Accordingly, our research focused on a practical examination of these issues, addressed to provide students with the ability to relate theory to practice (Bolanakis *et al.* 2008, 2009b).

The subsequent action in our research relied on the following decision. Despite the fact that students should obtain an appropriate and effective learning of the examined MCU architecture, they should also earn the benefit of learning how to apply their knowledge under any circumstances. Accordingly, our research focused on a systematic approach, addressed to help the student in bridging the gap between high- and low-level programming (Bolanakis *et al.* 2009c). This choice stemmed from the need to help students relating new information to their educational background and thereafter providing them with the ability of proceeding to a future self-study of other MCUs. It is worth noting that this initiative relied on a Complex Instruction Set Computer (CISC)-like assembly level programming due to the fact that it facilitates the passage from a higher to a lower level of programming (Bolanakis *et al.* 2009d).

The final decision for the theory class stemmed from the need to provide students with the solid fundamental knowledge, which is regularly missing in professional courses. Accordingly, we

proceeded to an overview of the fundamental fixed-point number representation and arithmetic theory in a systematic approach, which is subsequently applied to practicable set of examples (Bolanakis *et al.* 2009a). At this point it is worth noting that switching from a typical computer to microcomputer architecture may be provided to make the strategies more understandable to the reader. For example, a practicable set of arithmetical operations in assembly language gives students the way to (a) understand the assembly level techniques for supporting a wider range of values by means of a microcomputer system that is regularly addressed for a limited range of arithmetic operations and (b) exploit arithmetic theories so as to evaluate the advantages/disadvantages of the various signed representation systems in terms of the assembly code optimisation (Bolanakis *et al.* 2009a).

4. Activities for an effective cross-disciplinary research

Following our interdisciplinary research study, this section presents strategies for building bridges across engineering disciplines (with reference to professional courses) by means of a generalised working framework. The sequence of actions presented hereafter provides the necessary information for helping instructors focus on the successful and effective crossing of boundaries between two individual disciplines.

4.1. *Literature review*

An idea of a cross-disciplinary research on a course between two engineering disciplines, which is related to a more practicing approach in the former case (referred to as leading discipline/course) and a more scientific approach in the latter (referred to as migrating discipline/course), should be originated from a literature review. Instructors should study the historical advancement and current state of the course in both directions. Reviewing the literature would help the researcher who attempts cross-disciplinary collaboration (referred to as a promoter researcher) to examine and record advantages and disadvantages of this particular research. It is worth noting that the promoter researcher should be necessarily rooted in his/her epistemology (Borrego and Newswander 2008), that is, the leading discipline. It is also valuable for the promoter researcher to have been involved in the migrating discipline as well, since it would help him/her identify the significance of this endeavour.

4.2. *Cross-disciplinary collaboration support*

For successful cross-disciplinary research, it is strongly recommended that an interaction of the promoter researcher with at least three collaborators majoring in (1) the migrating discipline, (2) a resembling to the leading discipline, and (3) the discipline of didactics, is required. At first, collaborators should proceed in an exchange of information regarding each individual epistemology, in order to verify the worth of this particular cross-disciplinary research. Thereafter, collaboration between the promoter researcher and the researcher in the migrating discipline will clarify the demands of teaching in the migrating discipline and give way to the expected learning experience for the students. Collaboration between the promoter researcher and the resembling (to the leading discipline) researcher will give way to identify issues related to the encountered difficulties of the leading course, which are normally increased in the migrating course. Finally, collaboration between the promoter researcher and the didactics researcher will lead to solutions for overcoming comprehension difficulties encountered by the students.

4.3. *Laboratory class development*

With reference to the development of a professional course in the migrating discipline, the research should be initiated to address issues related to the laboratory training by emphasising on fundamental topics, while excluding too many technical details. A literature review of hands-on experiments education in allied disciplines would provide valuable feedback to the instructors. Proceeding to the implementation and subsequent evaluation of the laboratory class will clarify the encountered difficulties of students and orientate instructors' lines of research.

4.4. *Theory class development*

With reference to the migration of a professional course in a less practicable class, researchers should pay attention to the following. From the viewpoint of professional (practicing) courses, tutoring is limited to an exhaustive set of technical specifications and applications. Regarding the fact that professional courses are addressed to students with a sufficient educational background, the inquiry of technical specification during the practice addresses the students to an effective learning of the employed technology. However, the migration of a technological course to a more scientific level should be considered to provide students with the ability to transcend the limits between allied technologies, as well as exploiting their obtained experience in their future profession. Accordingly, instructors should proceed to educational methodologies and strategies that allow the crossing of boundaries between practicing and non-technology-related courses.

5. Conclusion and future directions

The article has presented an educational research on crossing boundaries between engineering disciplines successfully and effectively. The authors provided in detail, their long-term research in alternating the traditional computer architecture class to microcomputer architecture, which is presently experienced at the Department of Communications, Informatics and Managements, Epirus Educational Institute of Technology, Arta, Greece. This work considered and verified the possibility of extending a regular technological course to a more scientific level, and is addressed to provide a different aspect on professional courses (that are, nowadays, addressed at the expense of traditional courses). The authors' propositions are in line with a cross-disciplinary collaborating strategy for a successful building of bridges between science and technology in consideration of an effective professional course. The authors' future plans are orientated towards issues regarding the remote experimentation abilities of interdisciplinary courses.

References

Agarwal, K.K. and Agarwal, A., 2004. Do we need a separate assembly language programming course? *The Journal of Computing in Colleges*, 19 (4), 246–251.

Anguita, M. and Fernandez-Baldomero, F.J., 2007. Software optimization for improving student motivation in a computer architecture course. *IEEE Transactions on Education*, 50 (4), 373–378.

Bolanakis, D.E., Glavas, E., and Evangelakis, G.A., 2007a. A multidisciplinary educational board system for microcontrollers: considerations in design for technically accurate custom-made platforms. *Proceedings of the 1st international symposium on information technologies and applications in education*, November 2007, Kunming, P. R. China. IEEE Press, 391–395.

Bolanakis, D.E., Glavas, E., and Evangelakis, G.A., 2007b. Levin's approach for microcontrollers tutoring. *Proceedings of the 6th ASEE global colloquium on engineering education*, October 2007, Istanbul, Turkey. ASEE Press, 1–11.

Bolanakis, D.E., Glavas, E., and Evangelakis, G.A., 2007c. An integrated microcontroller-based tutoring system for computer architecture laboratory course. *International Journal of Engineering Education*, 26 (4), 785–798.

Bolanakis, D.E., *et al.*, 2008. Teaching the addressing modes of the M68HC08 CPU by means of a practicable lesson. *Proceedings of the 11th IASTED international conference on computers and advance technology in education*, September–October 2008, Crete, Greece. Acta Press, 446–450.
Bolanakis, D.E., Laopoulos, T., and Kotsis, K.T., 2009a. Fixed-point arithmetic for a microcomputer architecture course. *Computer Applications in Engineering Education* (submitted for publication).
Bolanakis, D.E., Kotsis, K.T., and Laopoulos, T., 2009b. Arithmetic operations in assembly language: educators' perspective on endianness learning using 8-bit microcontrollers. *IEEE 5th international workshop on intelligent data acquisition and advanced computing systems: technology and applications (IDAACS'2009)*, September 2009, Rende, Italy. IEEE Press (in production).
Bolanakis, D.E., *et al.*, 2009c. A teaching approach for bridging the gap between low-level and higher-level programming using assembly language learning for small microcontrollers. *Computer Applications in Engineering Education* (early view in advance of print).
Bolanakis, D.E., Kotsis, K.T., and Laopoulos T., 2009d. Teaching concepts in microcontroller education: CISC vs RISC assembly-level programming. *Proceedings of the international conference on information communication technologies in education*, July 2009, Corfu, Greece University of the Fraser Valley Press, 742–750.
Borrego, M. and Newswander, L.K., 2008. Characteristics of successful cross-disciplinary engineering education collaborations. *Journal of Engineering Education*, 97 (2), 123–134.
Buckner, K., 2006. A non traditional approach to an assembly language course. *The Journal of Computing in Colleges*, 22 (1), 179–186.
Burney, F.A. and Abdul Haq, A.K.M., 1991. Microcomputer usage by engineering students – a computerized survey. *European Journal of Engineering Education*, 16 (1), 85–94.
Culbreth, W.G., 2001. Meeting the needs of industry: development of a microcontroller course for mechanical engineers. *Proceedings of the 2001 ASEE annual conference & exposition*, June 2001, Albuquerque, NM. ASEE Press, 1–9.
Duntemann, J., 1992. *Assembly language: step by step*. New York: John Wiley & Sons Inc.
Giurgiutiu, V., Lyons, J., and Rocheleau D., 2004. Mechatronics/microcontrollers education for mechanical engineering students at the University of South Carolina. *Proceedings of the 2004 ASEE annual conference & exposition*, June 2004, Salt Lake City, Utah. ASEE Press, 1–10.
Hamblen, J.O., Parker, A., and Rohling, G.A., 1990. An instructional laboratory to support microprogramming. *IEEE Transactions on Education*, 33 (4), 333–336.
Hamrita, T.K., 2002. Micro-controllers in the biological and agricultural engineering curriculum at the University of Georgia. *Proceedings of the 2002 ASEE annual conference & exposition*, June 2002, Montreal, QC. ASEE Press, 1–6.
Hamrita, T.K. and McClendon, R.W., 1997. A new approach for teaching microcontrollers courses. *International Journal of Engineering Education*, 13 (4), 269–274.
Hazzan, T.S. and Karni, E., 2006. Similarities and differences in the academic education of software engineering and architectural design professionals. *International Journal of Technology and Design Education*, 16 (3), 285–306.
Levin, J.R., 1981. *On functions of pictures in prose*. New York: Academic Press.
Lodge, K., 2006. The programming of a micro-controller as the laboratory component in process control for undergraduates in chemical engineering. *Proceedings of the 2006 ASEE annual conference & exposition*, June 2005, Chicago, IL. ASEE Press, 1–12.
Margush, T.S., 2006. Using an 8-bit RISC microcontroller in an assembly language programming course. *Journal of Computing Sciences in Colleges*, 22 (1), 15–22.
Maurer, W.D., 2005. The effect of the Harvard architecture on the teaching of assembly language. *The Journal of Computing in Colleges*, 20 (5), 79–90.
Mayer, R.E. and Gallini, J.K., 1990. When is an illustration worth ten thousand words? *Journal of Educational Psychology*, 82 (4), 715–726.
Mengel, S.A. and Carter, L., 1999. Multidisciplinary education through software engineering. *Proceedings of the 29th ASEE/IEEE frontiers in education conference*, November, 1999, San Juan, Puerto Rico. IEEE Press, 13a3.12–13a3.17.
Montanez, E., 2005. Microcontrollers in education: embedded control – everywhere and everyday. *Proceedings of the 2005 ASEE annual conference & exposition*, June 2005, Portland, OR. ASEE Press, 1–10.
Osborne, H., 2002. The postroom computer: teaching introductory undergraduate computer architecture. *Proceedings of the 32nd SIGCSE technical symposium on computer science education*, February 2002, Convington, KY. ACM Press, 157–161.
Silver, J.L. Jr. and Leeper, R.R., 1983. Schemata for teaching structured assembly language programming. *SIGCSE Bulletin*, 15 (1), 128–132.
Sparks, J.J., 1993. Engineering education in a world of rapidly changing technology. *Proceedings of the AEESEAP/FEISEAP/IACEE international conference on engineering education*, November 1993, Singapore. 1–11.
Stallings, W., 2003. *Computer organization and architecture: designing for performance*. NJ: Pearson Educations Inc.
Tran, C. and Robillard, P.N., 1985. Teaching structured assembler programming. *SIGCSE Bulletin*, 14 (4), 32–44.
Wolfe, G.S., *et al.*, 2002. Teaching computer organization/architecture with limited resources using simulators. *Proceedings of 33rd SIGCSE technical symposium on computer science education*, February–March, 2002, Convington, KY. ACM Press, 176–180.

About the authors

Dimosthenis E. Bolanakis was born in Crete, Greece in 1978. He obtained a B.Sc. degree in 'Electronic Engineering' from the Department of Electronics, Thessalonikis Educational Institute of Technology, Greece, and a M.Sc. degree in 'Modern Electronic Technologies' from the Department of Physics, University of Ioannina, Greece. Presently, he pursues a Ph.D. degree at the Department of Primary Education, University of Ioannina, Greece. He has (co)authored more than 10 papers (mainly on research in education), while he has refereed articles for the IEEE Multidisciplinary Engineering Education Magazine and the International Journal of Engineering Education. He has been teaching Computer Architecture at the Department of Communications, Informatics and Management, Epirus Educational Institute of Technology, Arta, Greece for the past six years, while he has also participated in research projects for Reinforcing Informatics' Education and Broadening Higher Education. He is a student member of the IEEE Educational Society, ACM Special Interest Group on Computer Science Education and ACM Special Interest Group on Information Technology Education. His research interests include remote experimentation, teaching approaches for enhancing engineering education and the design of innovative educational hardware systems.

Konstantinos T. Kotsis received the B.Sc. degree in Physics Department from the Aristotle University of Thessaloniki, Greece in 1980 and his Ph.D degree from the University of Ioannina, Greece in 1986. In 1987, he joined the Department of Physics, University of Ioannina, Greece. He is now an Assoc. Professor at the Primary Education Department, University of Ioannina, Greece, and presently teaches the graduate elective course Didactics of Science in Primary Education and Department of Physics, University of Ioannina. His primary research interests focus on Didactics of Physics and Science using ICT.

Theodore Laopoulos is Associate Professor at the Electronics Lab., Physics Department, Aristotle University of Thessaloniki, Greece. His interests are in the fields of: Instrumentation Circuits and Systems, Sensor Interfacing Electronics, Measurement Techniques, Microcontroller Systems, and Development of Education in Electronic Instrumentation.

Dr. Laopoulos has published over 100 papers in international scientific journals and conferences, and has served as leader or senior researcher in more than 20 Greek and European research projects. Dr. Laopoulos is an IEEE senior member, Associate Editor of the IEEE Transactions on Instrumentation and Measurement, and chairman of the Advisory Board of 'IDAACS' - International Workshop on 'Intelligent Data Acquisition and Advanced Computing Systems'.