Assembly Language Curriculum Realignment in Computer Engineering at UCSC

Stephen C. Petersen, Alexandra Carey, Richard Hughey, David Meek
Department of Computer Engineering, University of California, Santa Cruz
petersen@soe.ucsc.edu

Introduction

Introduction to Computer Organization, numbered CMPE12C is the first lower-division computer-related course taken by most undergraduate students majoring in Computer Science (CS), Computer Information Systems (CIS), Electrical Engineering (EE) or Computer Engineering (CMPE) at the University of California, Santa Cruz. It teaches the functions and interrelations between the basic parts of computers and introduces assembly language to all students within the School of Engineering (except bioinformatics majors). Hence, effectively teaching the foundations of computer organization is important at this early stage for several reasons. First, students frequently solidify attitudes about whether or not to pursue fields requiring further study involving computers. Secondly, our experience in the classroom reveals that many of them find the hurdle of bridging the conceptual relationship between computer hardware and low-level programming very difficult to leap. This is not too surprising if we pause for a moment to gaze back at how computers have developed historically and note that a viable abstraction of the *actual hardware* is necessary and heuristic to real insightful understanding. Basic topics like assembly-language programming, registers as ports or memory, interrupts and virtual machines require a solid foundation built on simple initial ideas that later on can be understood with more depth and subtlety. Students need to form their own way of intuiting these things based on empirical knowledge formed as a consequence of programming real hardware they can interact with – not merely abstractions of computer-simulated hardware presented only as virtual machines or found opaquely within a complex workstation.

This paper discusses our attempt to address this problem by focusing on one critical aspect of how computer organization is taught; namely, on the way assembly language is discussed and intuitively understood by trying to make the relationship between a central processing unit (CPU) and its associated organized hardware tangibly and lucidly transparent to students. Prior to the revisions discussed below, the programming component of CMPE12C was taught on simulators and Personal Computers (PC's), a situation affording no satisfying view of the real underlying hardware inside. More often than not, students pondered the concrete more than the abstract – they wanted to know what was *really* going on "under the hood". Thus, our agreed upon primary goal was to essentially bring the hardware conceptually closer to students by *showing* them the excitement of programming hardware they could readily see, feel and

controllably understand by interacting with simple things, such as ports interfaced to light emitting diodes and input switches, interrupt buttons and liquid crystal displays. Feedback from students in the form of written course evaluations and instructor-student dialogs over the past several years have confirmed we were successful in meeting this goal.

A secondary goal was to revise the course curriculum to better serve the needs of later courses in microprocessor systems, computer architecture, compiler design, and operating systems. These subsequent classes have a broad range of needs. Computer Architecture focuses on pipelining and memory systems for RISC ("reduced instruction set computer") processors[1]. For this class, students need to know about memory, data types, low-level computer operations, and RISC assembly language (MIPS, in this case). Compiler Design requires essentially the same background but presently uses SPARC assembly language. Our Operating Systems course requires that students have a preliminary conceptual understanding of virtual memory and a strong understanding of interrupts. In Microprocessor System Design, we survey the engineering design of different classical bus architectures, and discuss various CPU's and peripherals. In the laboratory students's design and build their own custom microprocessor projects using the popular and versatile 68HC11, an embedded CISC ("complex instruction set computer") CPU. This laboratory particularly needs an adroit understanding of the HC11's software architecture, *i.e.* its unique assembly language. Giving students a better introduction to HC11 assembly language would allow us to concentrate more on hardware system architecture and advanced software programming techniques like mixed assembly and C, which has routinely been taught in the laboratory since 1995.

Based on these diverse needs, we took the unusual step of deciding to teach two assembly languages, MIPS and HC11, in the computer organization course. These would be treated in a newly created separate laboratory, thereby allowing us to meet both our primary and secondary goals. The work to design this laboratory divided naturally into two major tasks: development of the actual tutorial hardware to be used, and writing an accompanying lab manual to support both assembly languages, HC11 and MIPS. Thus, the hardware would employ the 68HC11 and be designed to meet the primary goal of bringing the hardware closer to the students while the lab manual meets both goals. The following two sections discuss details about changes we made to the course, and how and why the necessary hardware was designed and built here at UCSC. Although this was formally dubbed the Microcontroller Kit, it colloquially came to be known more widely as the "12C Microkit" or just "Microkit."

Making the Class and Laboratories

We should first note that CMPE12C additionally teaches critical skills needed to gain facility with arbitrary number bases and number representations, familiarity with bit-wise operations, understanding of memory layouts, the use of arrays and interrupts. Consequently, as part of this revision, we also changed the programming prerequisite from one quarter of C to two quarters of programming (now primarily in Java) to ensure that students coming into the class would have a more uniform background.

Based on the considerations noted above, our ideal textbook for the class would be one that discusses number and data representations, basic assembly language programming, interrupts, and introductory computer organization using as examples both the MIPS and the HC11 processors. It comes as no surprise that we were unable to find a text with this mixture, and routinely draw upon several resources throughout the intense 10-week quarter (with 3.5 hours of lecture and 2-4 hours of laboratory for week).

We begin the class using the SPIM simulator and the MIPS assembly language. We are presently using the text by Miller and Goodman[2], a relatively gentle introduction to computer organization and assembly language. This text, designed for a complete course, begins with a simplified assembly language which we don't use simply to avoid the unnecessary confusion learning three assembly languages would engender; two is quite enough. Using the simulator, students write the typical number conversion and data structure programs (static, queue, multi-dimensional array) or MIPS assembly language examples. The most important parts of this include becoming familiar with decomposing high-level concepts into individual machine instructions, and becoming completely familiar with binary and hexadecimal number representations. We frequently also discuss trinary and other bases in the class, just to ensure that everyone fully understands the underlying concepts.

About five weeks into the quarter, the lab and class change over to using and discussing the HC11-based Microkits, with the assistance of Motorola's donated manuals[3] and our own home-grown comprehensive laboratory manual[4]. This change in assembly language is coordinated with the class discussing procedure calls. Thus, students first see the MIPS calling conventions, in which the stack must be maintained by hand, and then seeing the more convenient (from the assembly language programmer's point of view) system of automatically placing the return address on the stack. The class then turns to interrupts, a topic that cannot effectively be taught with simulators. While the lectures study interrupt processing, storing registers and the like, students integrate multiple functions on the Microkit and use the external interrupt button to switch between them (Fig. 2). In the past, these simple exercises have been fraught with difficulty for many students, but now with the Microkit they have become a particularly effective learning tool; students quickly grasp the importance of enabling and disabling interrupts at appropriate times as well as understanding the inherent asynchronous nature of interrupts.

The laboratory in a course like this would generally be supervised and taught primarily by traditional graduate teaching assistants (TA's). We found experimentally that one of the most important features of the reorganization were the benefits of hiring many undergraduates as tutors to assist in teaching the laboratory, thereby lowering the student-to-instructor ratio. Under tutor supervision, students are able to discuss and question topics covered in class and then learn by discovering computer organization first-hand. Studying or developing low-level algorithms in small groups or with excited undergraduate tutors creates an atmosphere of teamwork and insight. Following these discussions, programming and debugging is done individually. The tutor-student relationship is also synergistic: tutors solidify their own understanding while benefiting their peers.

The first quarter using the new course layout quickly revealed the need for a good comprehensive laboratory manual. We consequently spent considerable time and effort creating a really good manual to be uniformly used in all laboratory sections. Obviously this enhances better continuity and consistency of treatment. For TA's, the manual also provides grading guidelines and candidate topics for discussions to be used by tutors. More time was found available to help students individually because the manual also provides tutorial help for common topics. The manual unifies the curriculum and makes the course easier to teach and administer, especially for new instructors. It was recently modified to reflect changes in the course as we moved to an open-source assembler.

Laboratory programming assignments are given electronically, typically once per week, over the course of ten weeks. Students spend four hours in scheduled sessions per week, but are given the option of working in the laboratory during other times as well. Assignments vary in difficulty and complexity, from basic exercises in efficiency in MIPS to programming intricate routines to handle external and internal interrupts with the Microkits. Each assignment also presents students with opportunities to excel by electively choosing to attempt some of the more difficult but otherwise optional features. Necessary algorithms and hints are provided both in the description of the assignment and in the lab manual. Assignments are submitted and graded electronically, but immediate feedback from tutors is also provided during students' scheduled laboratory sessions.

Typical laboratory assignments include MIPS efficiency, simple character I/O routines, self-modifying code in MIPS (replacing a line of code with another), an RPN calculator, a ROT-13 converter on the HC11, and some programming exercises in procedure calls, recursion, interrupts, and even binary search trees. All programs are written exclusively either in MIPS or HC11 assembly language.

Making the Microkit

Work began in earnest after receiving funding from the Campus and the School of Engineering, who initially split our $8,000 proposed budget for building 100 units. Of course, this was a true engineering experience, and the final cost was about $12,000, supported in part by a generous donation from Emeritus Professor Harwood Kolsky.

Initial discussions with faculty and technical staff quickly revealed that what we wanted probably didn't exist on the market, and a custom design was deemed too expensive. Rather than try and rely exclusively on professional consulting services or purchase an existing system, we decided, as a subordinate goal, to make the entire hardware phase of the project involve as many students as possible. In other words, let engineering students drive the design. Hence, this was subsequently offered as one of several candidate projects to the Fall Quarter session of CMPE123, one of our upper division capstone engineering design classes.

A team of four students chose to collaborate on the project. They first created a timeline and defined the usual industrial milestones to be met as the course progressed. Taught by Stephen

Petersen, the instructor acted as project engineer and mentor with the students serving in various engineering roles under him. The resulting division of labor had two students responsible for the hardware design and two students for the software design. Everyone participated in hashing out the overall system specification before beginning work on the prototype. This was an interesting experience since the School of Engineering was the client, and the students needed periodic consultations to confirm that what they were brainstorming was suitable, and affordable.

The team originally intended to base their design efforts on Motorola's classic expanded bus M68HC11EVB Evaluation Board.[5] They began by reverse engineering both its assembly code and circuit design, and quickly came to two conclusions: First, the EVB's use of available memory was insufficient as it implemented only a small portion of the available 64k. Since undergraduates were making something for their colleagues, they wanted to give them the best money could buy! Secondly, since the code was written exclusively in assembly language, student's found it primarily useful to understanding how the necessary context switch that would be needed to run downloaded programs could be accomplished, but was considered inappropriate for the ambitious and complex monitor program that was proposed. Thus, a decision was made to develop an entirely new system utilizing the full 64k memory, wasting as little as possible, and write the software primarily in C using assembly only where appropriate and necessary.

Basic features were then defined: the unit should have useful but intuitively simple I/O; a serial interface to a laboratory host computer; LCD display; an external interrupt capability; must be portable with SRAM battery backup; it should have an "open" hardware circuit board architecture that would afford students a real open "touch and feel" without being fragile; expansion bus capability for other possible uses than CMPE12C; it must be durable enough to withstand toting about by undergraduates for 10 weeks, and should also have a realistic target cost of about $100 for each unit. Initially the customer stated, "100 units would be sufficient". As a buffer for future attrition, 120 printed circuit boards were made during the printed circuit board (PCB) production run, and eventually 120 Microkits were fully constructed. This was a successful pilot project demonstrating the feasibility of student-designed printed circuit boards using state-of-the art CAD tools that could be routinely included in project classes. A commercial PCB house[6] specializing in "quick-turn" or "rapid-prototype" engineering boards was chosen for the first three prototype runs that produced boards in 24 hours at reasonable cost. Two boards were made each time – a requirement of the PCB house. The final run of 120 finished boards used another vendor[7] that best met our pricing goals. In each case, students were exposed to using commercial vendors to render standard design output files as a professionally prepared circuit board fully mimicking common industrial engineering practice. The final fully assembled cost per unit was $118.

Additional funding of $4,000 beyond the original $8000 mentioned earlier was sought to cover additional expenses, such as those incurred with the University's Machine Shop to cut and drill the thick plastic mounting hardware, and also to buy AC adapters and serial cables to connect the Microkit to the general purpose campus-wide (*i.e.* non-engineering) lab computers maintained and operated by the Computer And Telecommunications Services (CATS) staff. Moreover, both the CATS labs and our own undergraduate engineering labs use PC's running MS NT. We

sought and found a cost-effective cross-development package[8] that would run on Microsoft (MS) NT, allowing students in CMPE12C to both simulate as well as fully assemble their programs and download them over the serial link at 9600 bps in Motorola S19 format using MS NT's HyperTerminal. At the time of this writing we are also experimenting with an open-source assembler mentioned earlier that would enable home use of Microkits without separate software licenses.

Following is a specification summary of the resulting hardware.

- 68HC11A8, expanded bus with 8MHz crystal.
- 32kx8 of program and code space organized from 8000h to FFFFh. A single 128kx8 Flash EEPROM (in a PLCC-52 socket) is used and divided into four external jumper-selected pages of 32k each. The monitor program used in CMPE12C resides in page 0.
- 24k of fixed on-board user accessible static ram.
- One set of 8 SPST slide switches (*not* DIP switches) tied to an input port.
- One set of 8 red high intensity LED's driven by an output port.
- One 8-bit digital to analog port with two outputs: (1) drives an AC 500mW audio amplifier with potentiometer amplitude control. Speaker must be connected separately via a 2-pin header connection. (2) DC output extended from the DAC to a 2-pin header connection.
- A single 16 character by 2 lines LCD with intensity control.
- Single 8-bit unipolar DC analog to digital input port, 0 to 5.0V full range.
- Single 8-bit AC analog to digital input port: 50k input impedance, 4.5Vpp max.
- Jumper-selected AC or DC power source capability: (1) +7 to +25 VDC with reverse polarity protection, or (2) +7 to +25 VAC-RMS.
- Available on-board DC voltages: +5V(digital), +6V(analog), -12V(analog).
- Fully buffered expansion bus: –12V and +5V DC available; four partially decoded ports with 128 addresses each; first four bits of the address bus; two unbuffered A/D input ports, 0 to 4.5V full scale range; cpu embedded port-A (PA0-PA7) for access to input and output capture functions
- Single 9600 bps RS232c serial communications port.

The printed circuit board was fitted to the case and measures 5.65 x 8.0 inches (see fig. 1 below). All IC's are surface mount except those that students might in any reasonable way be able to electrically connect to. For example, this applies to the transceiver chips (74ACT244, 245) used with the bus expansion option, or several operational amplifier that interface the A/D and D/A features. All leaded through-hole components that could be wiggled, fatigued, and eventually broken were placed on the bottom of the PCB; this meant students had tactile access only to the top of the board, and was a specification of the design – make it "student proof". The one weakness here is the eight light emitting diodes driven by a memory-mapped output port that could, only with deliberate intent, be wiggled and broken. So far this has not happened. A later version of the PCB might use surface mount LED's instead. This was considered, but rejected at the time of design due to cost.

David Meek, a technical staff engineer associated within the Engineering School, provided invaluable advice throughout. He was enlisted to participate in helping us make final vendor/component selections and handle all purchasing. He also served as mechanical system integrator for the final top and bottom assemblies and latching case. This included mounting the PCB, working out details of how to "student-proof" the liquid crystal display (LCD) in the lid of the Microkit and helping with some form factor design issues, such as a robust chassis mounted into an aluminum case; these were incorporated into the layout of the final PCB revision so that a two-line alphanumeric display, serial cable, AC adapter, and expansion daughter cards would mesh nicely with the finished unit. Following completion of the class, several dedicated undergraduate volunteers did final production assembly and system testing. The first 60 kits were assembled, tested and ready for issue less than six months after the initial prototype design was done following the completion of the project class.

A total of 56 workstations in three CATS labs were initially outfitted to accommodate first use of the now christened "12C Microkit". Since the Microkit was portable, students could then use any of these workstations to conduct their labs. These were often shared with other courses and so were not always available to students. As a consequence of this and other mundane administrative and logistic issues, we deemed the periodic process of quarterly checkout and retrievals to be inefficient and time-consuming. Consequently, one of the University CATS computer labs located conveniently inside the Baskin Engineering building was selected for permanent installation as fixtures and has been in use since. There are now 27 permanent stations reserved exclusively for use by CMPE12C students in this lab. This has freed up the bulk of the kits to be made available to design courses where novel expansion features, such as Ethernet or wireless communications capability, can be incorporated into a standard reference base unit.

Several photographs of the finished Microkit are shown in Figures 1 and 2 following.

Fig.1. Finished Microkit showing significant features: LCD, slide switches, LED's, serial interface cable with DB25 connector, and protective plastic assembly.

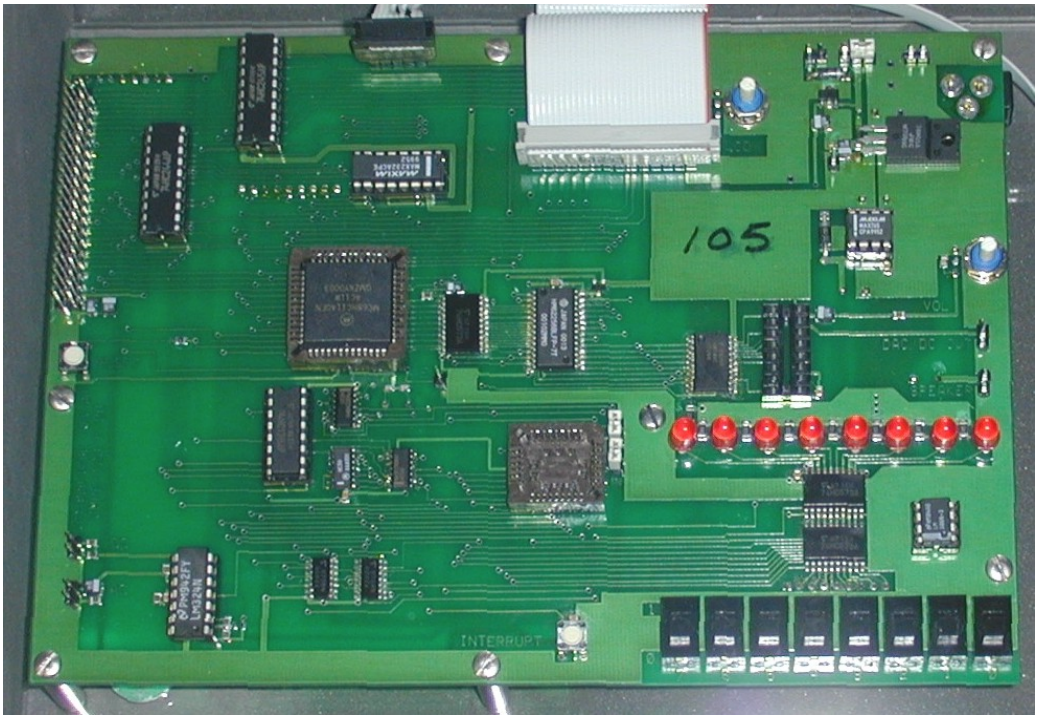

Fig. 2. Top view of the final PCB showing components. The slide switches and LED's are both mechanically and binary weight aligned (MSB is on the left). The white momentary pushbutton immediately to the left of the slide switches is a software debounced maskable external interrupt. The white momentary pushbutton on the far left edge is the system reset. A Type-1 external power plug can be seen in the upper right corner mounted with its three soldered through-holes. The two empty sockets house the ROM (PLCC) and DAC (DIP16).

Results

Prior to this revision, CMPE12C was relatively isolated in the curriculum; students learned Intel assembly language, unused in any of the following courses. The transition between the old curriculum and the new one was indeed bumpy, especially during creation of the new laboratory when the lab manual was being written. Undergraduates are enthusiastic about the use of the hardware, and now have far better preparation for courses that follow. Every quarter, several students ask if they can buy or build them. Microprocessor Systems Design has especially benefited since students now come into that class already knowing the tutorial processor's assembly language. Inclusion of the Microkit has also enabled advanced work in later project classes. The creation of the Microkit was itself a senior design project, and the kits continue to be used as the basis for new senior design projects. Last year, a student in CMPE123 successfully designed an Ethernet adapter that interfaced to the Microkit over its expansion bus. The small-group assembly language programming laboratories with their extensive tutoring support has also eased many students through the change from procedural C to Java - with its new object-oriented programming paradigm, which has many new foreign concepts. Overall CMPE12C now provides students a better more uniform background in the elements of computer architecture.

Several ideas are being entertained for the future. We are considering reversing the order of assembly languages in class. This would enable students to begin working immediately with the HC11 Microkits, seeing the hardware and making the lights blink early in the quarter. This would be followed by a transition to the RISC assembly language - so necessary for discussing modern pipelining and memory systems. After grasping the relationship to real hardware, the conceptual move to simulation of hardware becomes more understandable, albeit less interesting. We are also thinking about creating a stand-alone embedded systems course that would use the Microkits and associated peripherals for projects such as programming and audio synthesis. This would be a class primarily targeting computer science majors who do not wish to both build and program such a system.

Feedback from students and faculty confirm we were successful in realizing both our primary and secondary goals. Based on end-of-the quarter written evaluations, students have expressed an overall increased satisfaction with the course and its contents. Although certainly not earthshaking, compared to what we had before, this curriculum realignment has overall been a phenomenal success. Indeed, we consider this to be one of Computer Engineering's most effective curricular changes in recent years.

More information on the PCB hardware with a complete set of engineering schematics can be found at: http://www.soe.ucsc.edu/~petersen/microkit. The CMPE12C laboratory manual and course information can be found at http://www.soe.ucsc.edu/classes/cmpe012c.

Acknowledgments

In addition to the people mentioned above, this project would not have been possible without the support of the UCSC Committee on Teaching, the Baskin School of Engineering, a gift from Professor Emeritus Harwood Kolsky, Former Chair Joel Ferguson and Former Dean Patrick Mantey, graduate student and instructor Cliffton McIntire, graduate student David Dahle, and the many dedicated undergraduate laboratory tutors who have made this all work. Several undergraduate students, notably John Dempsey and Hugo Condeso responded to help us paint Tom Sawyer's Fence by volunteering to learn how to work with surface mount components and do the eternal and largely thankless task of final board assembly and system testing. The quality of the finished PCB's is due to solely to them.

We also particularly thank Motorola for their periodic donations each academic quarter of hundreds of HC11 manuals.

References

[1] John Hennessy and David Patterson, "Computer Organization and Design: The Hardware/Software Interface", second edition, Morgan Kaufmann, 1998
[2] Goodman and Miller, "A Programmer's View of Computer Architecture", Saunders College Publishing, 1993.
[3] M68HC11 Reference Manual, Rev 4, Motorola Inc. 2001 (supersedes older Rev3 which is no longer available);

M68HC11E Family Technical Data, Rev 3.
[4] A. Carey, C. McIntire, J. Ferguson, R. Hughey, "Computer Engineering 12C Lab Manual", 2000; http://www.soe.ucsc.edu/classes/cmpe012C
[5] M68HC11EVB/D1 Evaluation Board User's Manual, Motorola Inc.1986.
[6] Alberta Printed Circuits, Alberta Canada. Their proto-1 service has a reasonable flat fee and low additional cost/sq. inch; no silk-screen or solder mask; minimum of 2 panels per run; limited drill rack; http://www.apcircuits.com
[7] Advanced Circuits, Denver Colorado. An excellent cost-effective board house; http://www.4pcb.com/
[8] P&E Microcomputer Systems, Inc.: WinIDE11NT Editor/6811 Assembler, SIM11A 68HC11Ax Simultor; http://www.pemicro.com.

STEPHEN C. PETERSEN is a professional Consulting Electrical Engineer also teaching part-time as a Lecturer in Computer and Electrical Engineering. He received the B.S. and M.S. degrees, both in Electrical Engineering, from San Jose State University, and is a Registered Professional Engineer in the State of California. His consulting services include RF, analog, digital, programming and general R&D. He enjoys Amateur Radio, and holds an Amateur Extra Class license, call sign AC6P.

ALEXANDRA CAREY is an undergraduate majoring in computer engineering and mathematics. She has served as head tutor and as course assistant for CMPE12C several times. She is the primary author of the CMPE12C laboratory manual, which includes numerous examples on programming in assembly language. She is working with the Kestrel Parallel Processor team. Her interests include parallel processing, poetry, and swing dancing.

RICHARD HUGHEY received the B.A. in Mathematics and B.S. in Engineering from Swarthmore College, and the Sc.M. and Ph.D. in Computer Science from Brown University, and is Chair at the Department of Computer Engineering. His interests include parallel processing, bioinformatics, and curriculum development. His Kestrel Parallel Processor project has included over 20 undergraduate researchers.

DAVID MEEK is a Development Staff Engineer with the Engineering School. His primary job responsibility is to maintain, support and help with all aspects of the undergraduate Engineering Laboratories. Prior to coming to UCSC he worked 10 years as a senior engineering technician for Apple Computer in the Advanced Technology Group.