

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

Computer Organization is the first Computer Engineering course for most students at the University of California, Santa Cruz. It is frequently the point at which students decided whether or not to pursue computer engineering, computer science, or some other field. Prior to the revisions discussed below, the programming component was taught on simulators and PCs, a situation in which students gain no view or knowledge of the hardware inside. Our primary goal was to bring the hardware closer to the students, showing them the excitement of programming a portable embedded system with direct feedback, such as blinking lights and sounds. Our secondary goal was to revise the course curriculum to better serve the needs of the following courses in microprocessor systems, computer architecture, compilers, and operating systems.

The course, numbered CMPE12C, teaches the functions and interrelations between the basic parts of computers and introduces assembly language to all students within the School of Engineering (excepting bioinformatics majors). As with many such courses, some of the most critical skills to gain in this class include facility with arbitrary number bases and number representations, familiarity with bit-wise operations, and an understanding of memory layouts, the use of arrays, and interrupts. Prior to this revision, one quarter of programming (at the time in C) was the prerequisite. As part of this revision, we change this to the first two quarters of programming (now primarily in Java) to make more uniform the backgrounds of the students. We have frequently discussed having Discrete Mathematics (which also discusses number bases and basic logic) as a prerequisite, but this would make course selection exceedingly difficult for entering junior transfer students.

The subsequent courses have a broad range of needs. Computer Architecture focuses on pipelining and memory systems for RISC processors¹. For this class, people need to know about memory, data types, low-level computer operations, and RISC assembly language (MIPS, in this case). In Compilers, students need to know many of the same issues, but presently use SPARC assembly language. In the Operating Systems course, students need to have a preliminary understanding of virtual memory and a strong understanding of interrupts. In Microprocessor System Design, students build a 68HC11-based microprocessor system, and thus need an understanding of HC11 architecture and assembly language, and interrupts.

Based on the diverse needs of the following courses, we took the unusual step of deciding to teach two assembly languages, MIPS and HC11, in the computer organization course at the same time as we introduced a separate laboratory section (as opposed to on-your-own programming assignments) and our new HC11 Microkits. Prior to this revision, CMPE12C was relatively isolated in the curriculum; students learned Intel assembly language, unused in any of the following courses.

To begin the work, we received 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.

Although certainly bumpy, especially in determining the transition between the two assembly languages, the course revision has been a phenomenal success. Undergraduates are enthusiastic about the use of the hardware, and have far better preparation for all of the following courses. 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. The small-group assembly language programming laboratories, and their extensive tutoring support, has eased many students through this new programming paradigm, which has even more foreign concepts since the change from C to Java in the introductory programming courses.

In the remainder of this paper, we discuss the design and implementation of the Microkits, the course, and laboratories.

Making the Microkit

CMPE12C's most direct successor is the Microprocessor System Design class, CMPE121. Since 1995, this course has successfully used the Motorola 68HC11 microcontroller as the basis for circuit boards that are individually designed and hand wire-wrapped by each student in the laboratory phase of the class. As noted above, continuity with this course was the basic reason for choosing the same CPU for the new hardware. This way students would already have a thorough introduction to the HC11's assembly language, allowing course content in CMPE121 to shift and concentrate more on hardware system architecture and advanced software programming techniques like mixed assembly and C, which is routinely taught in the laboratory. Although the new hardware was formally dubbed the Microcontroller Kit, it colloquially came to be known more widely as the "12C Microkit" or just "Microkit."

Collaboration 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. Thus, 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 the 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. Four students undertook this task and finished the class with a working prototype. David Meek, technical staff engineer in the Engineering School, was enlisted to participate in helping us make final vendor/component selections and handle all purchasing, as well as to act as system integrator for the final top assembly of the completed Microkit. Following completion of the class, several dedicated undergraduate volunteers did final production assembly and system testing.

A team of four students in the Fall session of CMPE123 chose to collaborate on the project. They 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 in a non micro-managed environment. The resulting division of labor had two students primarily 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 Department was the “customer”, and the students needed periodic consultations to confirm that what they were brainstorming was suitable and affordable.

First, basic features were defined: the unit should have useful but intuitively simple I/O; a serial interface to a laboratory host computer; an 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 PCB production run, and eventually 120 Microkits were fully constructed. This was a successful pilot project for PCB CAD. A commercial PCB house² specializing in quick-turn engineering boards was chosen for the first three prototype runs that turned in 48 hours at reasonable cost. Two panels were made each time. The final production run of 120 finished boards used another vendor³ 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 total 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 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 NT. We sought and found a cost-effective cross-development package⁴ that would run on 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 NT's HyperTerminal. At the time of this writing we are also experimenting with an open-source assembler that will enable home use of Microkits without separate software licensing.

The team originally intended to base their design efforts on Motorola's classic expanded bus M68HC11EVB Evaluation Board⁵. 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 since it implemented only a small portion of the available 64k. Since undergraduates were making something for their colleagues, they wanted to give them the most bang for the buck! Secondly, since all the code was written in assembly language, it was primarily useful to understand how the necessary context switch that would be needed to run downloaded programs could be done, 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 map, write the software primarily in C and use assembly only where appropriate and necessary.

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 +15 VDC with reverse polarity protection, or (2) +7 to +15 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;

The printed circuit board was fitted to the case and measures 5.65 x 8.0 inches. All IC's are surface mount except those that students might in any reasonable way be able to connect to. For example, this applies to the transceiver chips (74ACT244, 245) used for the bus expansion option and op-amps 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.

Advice from technical staff proved invaluable. Input on some form factor design issues, such as a robust chassis mounted into an aluminum case, 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. The first 60 kits were assembled, tested and ready for issue less than six months after the initial prototype design was formulated.

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.

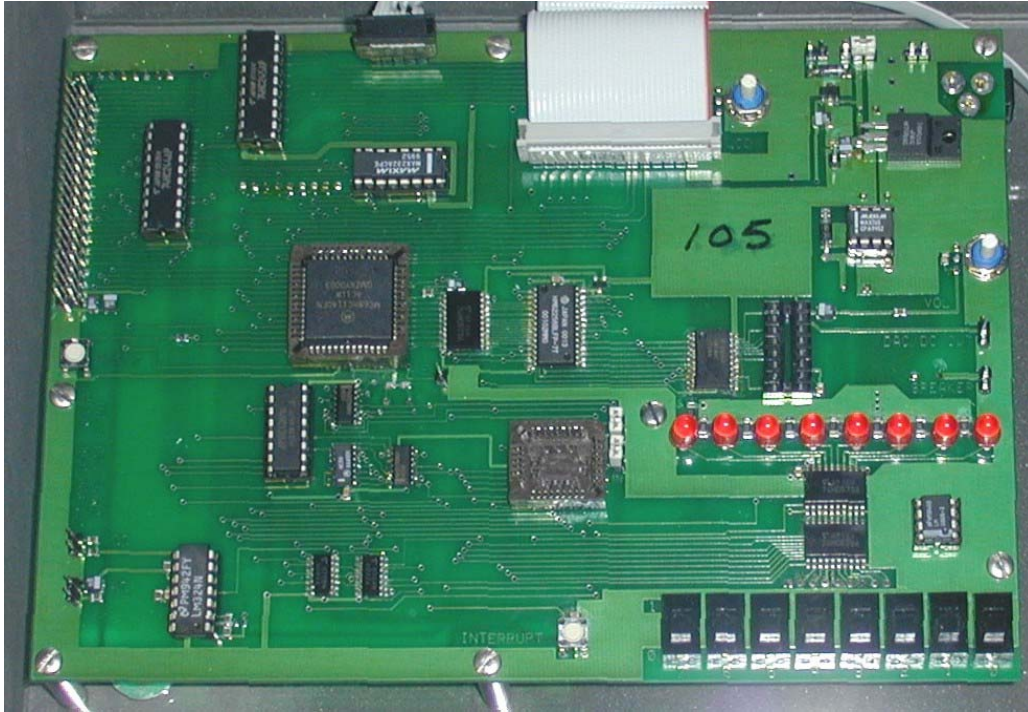


Fig. 2. Top view of the final PCB showing components. The slide switches and LED's are both mechanically and binary weight aligned (MSB is on the left). The white momentary pushbutton immediately to the left of the slide switches is a software debounced maskable external interrupt. The white momentary 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).

Making the Class and Laboratories

Our ideal text for the class would be one that discussed number and data representations, basic assembly language programming, interrupts, and introductory computer organization trying for example supposed the MIPS and the HC11 processors. Surprisingly, we were unable to find a text with this mixture, and use 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 Goodman⁶, a relatively gentle introduction to computer organization and assembly language. The text, designed for a complete course, begins with a simplified assembly language which we have dropped from our CMPE12C because we do not want to have the students becoming confused the introduction of three separate assembly languages; two is quite enough. Using the simulator, students write the typical number conversion and data structure (static, queue, multi-dimensional array) to assembly language examples. The most important parts of this include becoming familiar with breaking 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 class, the laboratory changes over to using the HC11-based Microkits, with the assistance of Motorola manuals⁷ and a comprehensive laboratory manual⁸. 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. Then, the class 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). These simple exercises are fraught with difficulty for many students, and have been a particularly effective learning experience where they quickly grasp the importance of enabling and disabling interrupts at appropriate times.

One of the most important features of the class has been hiring many undergraduates to assist in teaching the laboratory and provide students with one-on-one interactive help. 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 overexcited tutors creates an atmosphere of teamwork and insight. Following this discussion, programming and debugging is done individually.

The tutor-student relationship benefits the tutors as well. Tutors solidify their knowledge while helping their peers. Most go on to tutor other courses; some have become undergraduate course assistants or graduate student teaching assistants.

The need for a good laboratory manual quickly became evident. We spent considerable time and effort to create one for use in all of our laboratory sections. Tutors' laboratory sections became more uniform. For teaching assistants, the manual provides grading guidelines and discussions topics for tutors. Tutors found they had more time to help students individually, as the laboratory manual provides help for common topics. The manual also unifies the curriculum between different instructors. Recently, the laboratory manual was modified to reflect changes in the course as we moved to an open-source assembler.

Laboratory programming assignments are given electronically, typically one 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 on the HC11. Each laboratory assignment presents students with opportunities to excel if they choose to attempt some of the more difficult but optional features. Necessary algorithms and hints are provided both in the description and the lab manual. Assignments are submitted and graded electronically, but immediate feedback from tutors is also provided during students' scheduled laboratory sessions.

Typical laboratories 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 in assembly language.

Results

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. However, these workstations were shared with other courses and were not always available, and the process of checkout and retrieval of the kits each quarter eventually proved to be too time consuming, so one of the University CATS computer labs in the Baskin Engineering building was selected for their installation as fixtures and has been in use since. There are now 27 permanent lab stations reserved 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.

The curricular changes in Computer Organization have brought a new enthusiasm to freshman engineering students. It has enabled more advanced projects in microprocessor system design, and has made uniform the backgrounds of students in computer architecture. Overall, we consider this to be one of Computer Engineering's most effective curricular changes.

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. We would then transition to the RISC assembly language so necessary for discussing modern pipelining and memory systems. Of course, the unfortunate consequence of this reordering would mean having to motivate students to switch from real hardware to a simulator.

We are also considering the creation of a stand-alone embedded systems course that uses the Microkits and associated peripherals for projects such as programming and music synthesizer. This would be a class primarily targeting computer science majors who do not wish to both build and program such a system.

More information on the Microkit can be found at: <http://www.soe.ucsc.edu/~petersen/microkit>. Among other things, this site includes: Example HC11 assembly language programs; listing of the API "include" file student's need to access basic drivers and services, like the serial port, LCD ports, memory map of interrupt hooks and other useful procedures found in the Microkit monitor code; a complete set of engineering schematics. 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 School of Engineering, a gift from Professor Emeritus Harwood Kolsky, Former Chair Joel Ferguson and Dean Patrick Mantey, graduate student and instructor Clifton 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.

- [1] John Hennessy and David Patterson, "Computer Organization and Design: The Hardware/Software Interface", second edition, Morgan Kaufmann, 1998
- [2] 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>
- [3] Advanced Circuits, Denver Colorado. An excellent cost-effective board house; <http://www.4pcb.com/>
- [4] P&E Microcomputer Systems, Inc.: WinIDE11NT Editor/6811 Assembler, SIM11A 68HC11Ax Simulator; <http://www.pemicro.com>.
- [5] M68HC11EVB/D1 Evaluation Board User's Manual, Motorola Inc. 1986.
- [6] Goodman and Miller, "A Programmer's View of Computer Architecture", Saunders College Publishing, 1993.
- [7] M68HC11 Reference Manual, Rev 4, Motorola Inc. 2001 (supersedes older Rev3 which is no longer available); M68HC11E Family Technical Data, Rev 3.
- [8] A. Carey, C. McIntire, J. Ferguson, R. Hughey, "Computer Engineering 12C Lab Manual", 2000; <http://www.soe.ucsc.edu/classes/cmpe012C>

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.