

## **AC 2007-2683: A TWO-COURSE SEQUENCE IN COMPUTER ENGINEERING PRINCIPLES FOR ELECTRICAL ENGINEERING STUDENTS**

### **Dorin Patru, Rochester Institute of Technology**

Dr. Dorin Patru was born in Sibiu, Romania. He attended the local German school system through the 12th grade. He received both his BS and MS degrees in electrical engineering from the Technical University of Cluj-Napoca in Romania in 1993. From 1993 to 1995 he worked for the Institute for Design and Research of Automation Control and Test Equipment, located in Cluj-Napoca, Romania. In 1995 he joined the faculty at the Technical University of Cluj-Napoca as an Instructor and Research Assistant. In 1997 he received an assistantship for doctoral studies in the United States from Washington State University located in Pullman, WA. He received his Ph.D. in electrical engineering in the spring of 2002. His advisor was Dr. Scott R. Hudson and his thesis is entitled: "Application specific performance enhancements of digital and mixed-signal CMOS circuits using opto-electronic circuit techniques". Upon graduation, Dr. Patru joined the faculty of the Electrical Engineering Department at the Rochester Institute of Technology (RIT) as a tenure track Assistant Professor. He is currently teaching courses in the areas of: digital circuits and systems design, computer architecture and physical implementation of integrated circuits. He is also leading Project METEOR at the Rochester Institute of Technology, a sequence of multidisciplinary senior design projects.

### **Daniel Phillips, Rochester Institute of Technology**

Daniel B. Phillips was born in 1956 in Rochester, N.Y. He received a Bachelor of Science in Electrical Engineering in 1979 from the State University of New York at Buffalo where he continued graduate study in electrophysiology until 1981. He was employed in both the clinical and industrial sectors between 1982 and 1992 in the areas of automated test, embedded systems and biomedical data acquisition and control. After spending two years as a consultant to the Department of Anesthesiology at Yale University, he was accepted into a graduate course of study focusing on biomedical ultrasound at the University of Rochester in 1992 and received his Ph.D. in Electrical Engineering in 1998. He served as a scientist and an assistant professor of research in the Diagnostic Ultrasound Laboratory of Dr. Robert C. Waag at the University of Rochester from 1998 until 2000 at which time he was accepted into a tenure track teaching position in the Electrical Engineering Department at the Rochester Institute of Technology where he received tenure and a promotion to Associate Professor in 2006. His interests include biomedical applications of electrical engineering including signal processing and embedded systems. He is a principal faculty member associated with the establishment of the Biomedical Option in Electrical Engineering at the Rochester Institute of Rochester. He currently serves as the advisor for a number of engineering oriented student clubs as well as the Chair of the Rochester Section of the IEEE Engineering in Medicine and Biology Society.

### **Eric Peskin, Rochester Institute of Technology (COE)**

Eric Peskin is an Assistant Professor in the Electrical Engineering Department at the Rochester Institute of Technology. Prior to that, he worked as a Senior Automation Engineer in Logic Technology Development at Intel Corporation. He received the B.S.E. degree in electrical engineering from Princeton University in 1994, and his Ph.D. degree in Computer Science from the University of Utah in 2002. His research interests are in the areas of reconfigurable computing, asynchronous circuit design, nanotechnology, and computer aided design (CAD) / electronic design automation (EDA). He is a member of the American Society for Engineering Education (ASEE), the Institute of Electrical and Electronic Engineers (IEEE), the IEEE Computer Society, the Association of Computing Machinery (ACM), and the ACM special interest groups (SIGs) on Architecture (SIGARCH) and Design Automation (SIGDA).

# **A Two Course Sequence in Computer Engineering Principles for Electrical Engineering Students**

## **Abstract**

Traditionally computer architecture courses emphasize either a programmer's or logic designer's perspective with regard to computer engineering. Recognizing the value of both approaches, a sequence of two mandatory courses has been developed that addresses both of these aspects of computer engineering for the curriculum in Electrical Engineering at the Rochester Institute of Technology.

The lectures of each course are complemented by weekly lab sessions, in which the students complete assignments of increasing difficulty. In the labs associated with the course which emphasizes the programmer's perspective, students create assembly language programs which transfer and manipulate data, handle interrupts and use different peripherals available on a commercially available microcontroller. In the final two week lab assignment they create a rudimentary data acquisition and control system. In the labs associated with the course taught from a logic designer's point of view, students design, simulate and ultimately implement a personal, full custom microcontroller in a reconfigurable logic device. All assignments are hands-on and require physical demonstration of the working code and hardware.

In this paper, the motivation for the creation of this two course sequence is first presented. Then the topics covered by each course are outlined and how these topics meet the intended instructional objectives is shown. A description of the lab assignments, which complement the lectures and further foster the instructional objectives follows. Finally, possible future improvements are indicated.

## **Introduction**

The introduction of Very Large Scale Integration (VLSI) devices in the 1980s made the integration of memory and input / output peripherals along with the central processing unit possible. This resulted in the development of the prototypical microcontroller, also commonly referred to as an integrated microcomputer. Their ubiquitous use in almost all contemporary electronic systems indicates the importance of courses which teach electrical engineering students how to use and/or design microcontrollers. The intelligent use of microcontrollers implies knowledge and understanding of their resources and instruction set, and assembly language level software design. The design of microcontrollers entails computer architecture and logic design, two design domains which have recently been blended by Very Large Scale Integration.

To address these needs, the electrical engineering curriculum at the Rochester Institute of Technology includes two mandatory courses: "Microcomputer Systems - 0301-365" and "Computer Architecture - 0301-347". Both courses have evolved since they were introduced in the curriculum. "Microcomputer Systems" has today a strong emphasis on the interface between a microcomputer and its peripherals, and its lab content was significantly revised two years ago

to reflect this. “Computer Architecture” currently incorporates a blend of computer architecture and logic design. An associated lab for this course was introduced in the fall of 2003 in which students are guided through a bottom-top design activity that results in the implementation and testing of a complete microcontroller of simple complexity that is emulated in a reconfigurable logic device.

The paper first presents each course and associated lab content. Then it continues to show how this two course sequence serves in conveying computer engineering principles to electrical engineering students. Finally, possible future improvements are indicated.

### **The “Microcomputer Systems” course and its lab**

In this course the elements of a microcomputer are presented, including a detailed discussion of the memory, input-output, the central processing unit (CPU) and the busses over which they communicate. Assembly language level programming is introduced with an emphasis on enabling manipulation of elements of a microcomputer system using software. Efficient methods for designing and developing assembly language programs are presented. Concepts of program controlled analog and digital, input and output are studied in detail and reinforced with extensive hands-on lab exercises involving both software and hardware.

A student who successfully fulfills the course requirements will:

- Comprehend and explain the elements of a microcomputer system and their relationship to each other.
- Apply efficient methods to program a microcomputer system in assembly language, in a manner that allows the manipulation of hardware via software.
- Comprehend and explain the operation and relationship of software tools necessary to provide a development environment for a microcomputer based system.
- Design, implement, test and document an assembly language program to achieve a specified functionality.
- Understand and demonstrate how to incorporate an interrupt mechanism in software and hardware on a microcomputer system to achieve program operation that can respond to asynchronous events.
- Comprehend and explain the basic relationship between assembly language and the high level language C for programming a microcomputer based system.

### Course History and Revision:

The CPU model utilized for this course for a number of years was the Motorola 68000, a 16-bit microprocessor capable of 32-bit operations with multiple internal registers, a fairly comprehensive instruction set and a prototypical interrupt handling capability. This processor architecture represented a major step in computing systems and was incorporated in workstations, personal computers (the original Macintosh) and even hand-held calculators (the Texas Instruments TI-89). In fact some of the original laboratory experiments for this processor were based on Macintosh personal computers as development systems. Given that the 68000 is a microprocessor, hardware experiments required the utilization of a development board that incorporated memory, peripheral devices and interface circuitry. An Arnewsh SBC68K

evaluation board served as the foundation of the laboratories previously and the course revolved around text books focused on the 68000 architecture and assembly language software<sup>1,2</sup>. The labs focused mainly on assembly language programming skills and the use of a parallel port interface device to enable keypad scanning and multiplexed operation of 7-segment LED Displays, both in a polled mode of operation. These labs were revised somewhat to also include an introduction to serial communication and hardware interrupt functionality.

While the 68000 represents a prototypical CPU architecture and the laboratories demonstrated the integration of separate peripheral interface and memory devices, for practical reasons, the students were presented with a fixed microcomputer configuration and really only asked to observe the overall functionality of programs that were written (user input and output). A realistic constraint which prevented a more detailed interrogation of relevant system signals and possible reconfiguration of peripheral access was the fact that there was only a limited complement of the 68000 development boards (approximately 20 units) that had to serve a yearly class size of approximately 100 students. This limitation, along with the replacement cost, in terms of time and money (~\$500/board) made it mandatory that the hardware development boards were only accessible during limited lab periods (nominally three hours per week) unless special arrangements were made with department staff. Although simulator software that ran on personal computers was available, this arrangement frequently made the lab exercises and demonstration sign-offs a stressful experience which did little to encourage students in pursuing advanced courses in this area.

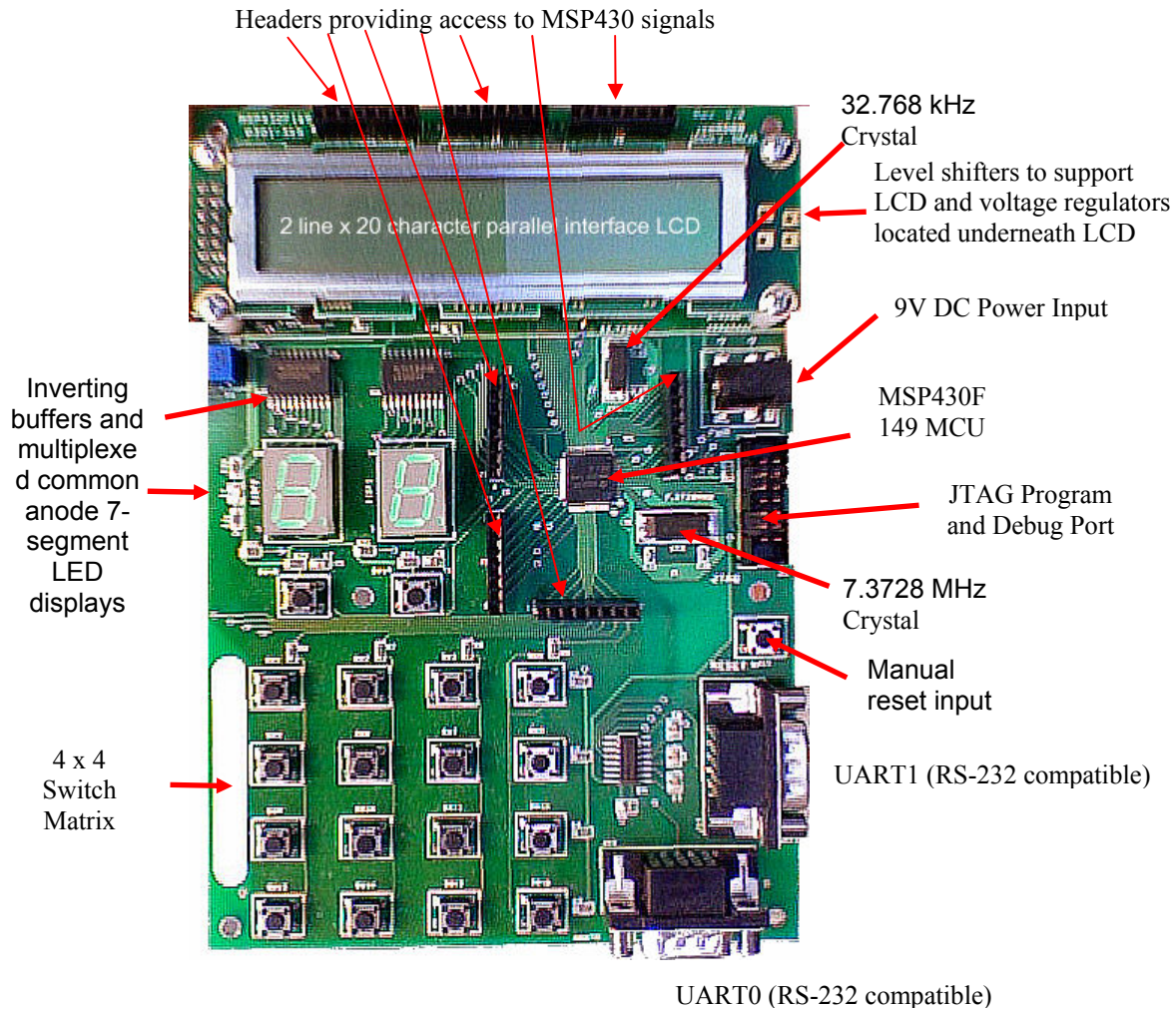
One of the advanced courses in the area of specialization is a course in embedded microcontroller system design (0301-644). In this upper-level, professional elective course, the students are exposed to the architecture and utilization of at least two microcontroller devices and expected, through a series of progressively more involved lab exercises to implement a multiple microcontroller project by the end of the course. To facilitate this effort, the students were required to purchase relatively inexpensive evaluation boards of two different microcontroller families (for example a Microchip PIC and a Zilog Z8) that can be purchased for under \$50 and a paper back text that was somewhat generic in terms of microcontroller utilization or systems design, once again for under \$50 representing a cost to the student of under \$150 for the course, the cost, in a number of cases, less than that of a conventional hard-cover text book. The development software that came with the evaluation boards was easily installed on a student's personal computer and also on the laboratory computers in the department. The students were provided with 24 hour, 7 day a week access to laboratories equipped with personal computers and test and measurement instrumentation (oscilloscope, triple DC power supply, function generator, digital multimeter) and a small locker for storage. This enabled the assignment of project-like laboratory assignments that were considered analogous to conventional homework assignments. The assignments were progressively more involved and built upon functionality of previous assignments. It was found that the students would work on these assignments on schedules that they arranged to meet specified completion deadlines. This allowed the students more opportunity to explore the operations and functionality of the microcontroller systems in more depth as well as the emphasizing the importance of time management. It also provided a learning experience that more closely reflected the actual activities and responsibilities of a practicing engineer, which since Rochester Institute of Technology has a very strong co-operative employment program, the students were very much

aware of. The student feedback from this course reflected the challenges associated with this type of project oriented laboratory experience as well as the benefits of more self-directed and organized design activities.

Based on the results and observations from the advanced microcontroller system design course, it seemed reasonable that the core Microcomputer Systems course might benefit from the same type of laboratory experience. A proposal was made for funding support from the Rochester Institute of Technology Provosts Learning Initiative Grant (PLIG) program to develop a custom board with a design that was at once generic and also provided suitable capabilities to address the course content and objectives of this lower level (typically taken in the Spring quarter of the second academic year) mandatory course.

The proposal was accepted and funded to manufacture a set of 100 boards based on the Texas Instruments MSP430 microcontroller family with a target cost of approximately \$100 including power supply and JTAG based programming interface. As shown in Figure 1, in addition to the microcontroller, it also provided two serial ports, seven segment LED displays, a two-line, 40 character LCD display and 18 discrete momentary switches for user interaction. The particular microcontroller model chosen also provided parallel input/output ports, timer/counters and analog-to-digital conversion capability along with 60 kilobytes of flash memory and 2 kilobytes of random access memory. While the choice of the microcontroller to be used as a case study is certainly open for debate, the Texas Instruments MSP430 family<sup>3</sup> was chosen for the following reasons:

- implementation of a true Reduced Instruction Set Computer (RISC) architecture
- existence of a nominally orthogonal instruction set including the availability of hardware multiply instructions as well as emulated instructions
- availability of wide variety of common input and output peripherals
- multiple interrupt capability associated with peripheral and CPU operation
- existence of multiple free and open source development environments available that students can install on their own personal computers to complete assignments and investigate concepts presented in the course
- existence of reasonably priced prototyping and development systems that an individual student can purchase in lieu of a conventional text (which is supplanted by handouts and on-line material and references).



**Figure 1 RIT MSP430 Development Board indicating salient features. The board was designed by Jason Mann while an Electrical Engineering student and manufactured by FSI Systems of Farmington, N.Y.**

In addition to these capabilities, the board was designed specifically to give the student access to all of the processor signal lines through the use of standard header sockets. The board design, test and verification were carried out by an upper level electrical engineering student and initial prototypes were assembled at Rochester Institute of Technology's surface mount facility in the College of Engineering. Another upper level student was responsible for refining and testing a series of lab experiments that closely matched the course content. Software development environments have included both GNU based tool suites running under Cygwin and Texas Instruments Eclipse-based Integrated Development environments. After testing and verification of the prototype designs, design refinement, testing and full assembly were contracted to a local engineering firm (FSI Systems of Farmington, New York) whose owner is a Rochester Institute of Technology Electrical Engineering alumni and where the original designer of the boards obtained full time employment. The systems were sold to the students through the Rochester

Institute of Technology bookstore for a cost of approximately \$170 after inclusion of non-recurring engineering costs and administrative overhead. While this seems slightly expensive, the revised course required no text and course content was provided via handouts and online reference material, which is very similar to the experience that a practicing engineer would face in working with current microcomputer systems and components. In addition, a significant feature of the design was that it would be flexible and capable enough for students who pursued advanced courses in embedded design would be able to utilize the same board. This was the case for the previously mentioned Embedded Microcontroller System Design as well as a graduate course entitled Advanced Topics in Embedded System Software Design (0301-742) in which the MSP430 based development board was used as a framework to investigate the design of an embedded real-time operating system and its application. It was also possible for the boards to be passed on to successive students taking this course at a cost decided upon by the student who originally purchased the board, usually at a discount significantly less than would typically be offered, for example, in the way of a text book buy-back program.

Table 1 outlines the topics covered in the lecture and in the lab. As can be seen, the lecture content proceeds from basic microprocessor architecture through assembly language programming and in to utilization of peripherals. The laboratory experiments follow this progression and become incrementally build upon topics and results that are obtained, ultimately resulting in a functional microcontroller based data, control and acquisition system that provides tangible and functional user interaction.

In the first seven labs, i.e. week two through eight, students learn how to use the MSP430 Microcontroller to perform different data transfer and manipulation tasks using CPU registers, Memory and Input/Output Peripherals. In the concluding two lab sequence they capitalize on that knowledge, and are strongly encouraged to REUSE their previous software efforts to design a stand alone Environment Monitoring System. The system has to acquire and display three different quantities: temperature, pressure and relative humidity. These are measured using sensors which output signals proportional to the quantity measured. Three keys are used to select the quantity that will be displayed on two 7-segment displays. The system samples each of the three input channels every 1 second, i.e. with a sampling frequency of 1Hz. As can be inferred, the system is in part similar to the home thermostat.

Since the students own the boards and have access to the development software, there is a great more flexibility in their efforts to complete the lab assignments. There are scheduled weekly three hour lab sessions that provide access to the faculty member and teaching assistants as well as the opportunity to demonstrate successful completion of the assigned tasks. The students are also granted access to laboratory facilities and instrumentation outside of the scheduled lab sessions. This has achieved the objectives of allowing more involved lab experiments as well as reducing some of the stress associated with accomplishing the required activities. Having more flexible access also allows opportunity for more flexible interaction between students, faculty and teaching assistants in that is more feasible to schedule discussion outside of the scheduled lab sessions.

During the second year that the course was offered, the students had the option of either purchasing the Rochester Institute of Technology development boards or a Texas Instruments Flash Emulation Tool (FET) that consisted of a printed circuit board with zero insertion force socket and header connections as well as a USB based JTAG programming interface. If they chose the FET, they were provided with the necessary peripheral devices (7-segment LED displays, hex keypads, etc.) to implement the necessary systems necessary for the laboratory experiments. Even though the FET option represented a significant savings, student feedback and evaluations revealed a preference for the Rochester Institute of Technology development board since it allowed them to focus on the programming and system level objectives as opposed to the hardware aspects of component integration. Student and faculty feedback has also provided impetus to initiate design refinements which might lower the cost of the board to the students. Experience with both the MSPGCC open source development suite and the Texas Instruments Code Composer Essentials IDE has motivated the consideration of developing a less expansive and more directed set of software tools for utilization at this level of student experience, i.e., the flexibility and sophistication of both environments can appear to be somewhat overwhelming to the student and can make the subject matter appear to be needlessly complex.

As originally conducted the course met four times a week for 50 minutes each class. An administrative decision resulted in that being reduced to three times a week for 50 minutes each class. Most recently, the contact hours associated with this class were returned to the original four 50-minute classes per week. The topic coverage based with this number of contact hours per week is shown in

Table 1.

Week	Lecture	Lecture Topic	Lab Topic
1	1	Syllabus    Introduction to the Microcomputer Block Diagram.	No Lab during the first week.
	2	A Microcomputer based Data Acquisition and Control System example – The Home A/C System.	
	3	Flow Charts – Algorithmic State Machine Charts.	
	4	The Fetch – Decode – Execute Cycle.	
2	5	The Instruction Set Architecture – The Instruction Set.	Introduction to the Code Composer Essentials (CCE) Integrated Development Environment (IDE).
	6	Resources: Registers and Memory Organization	
	7	The Instruction Word	
	8	Assembler – Linker and Machine Code Addressing Modes – Register Mode	
3	9	Addressing Modes – Indexed, Absolute and Immediate	Introduction to Assembly Language Programming. Assembler directives. Bit manipulations.
	10	Addressing Modes – Symbolic Mode	
	11	Addressing Modes – Indirect Register and Indirect Auto-Increment.	
	12	Test 1 Review.	
4	13	<b>Test 1</b>	Introduction to Assembly Language Programming – continued. Array manipulations.
	14	Digital Input / Output.	
	15	Interrupt basics	
	16	Simple Interrupts.	
5	17	Hardware Counters Review.	Digital Input / Output Polling. Multiplexed writing to 7-Segments Display.
	18	Timer / Counters Features, Hardware and Operation Modes.	
	19	Timer / Counters Operation Modes cont. Capture / Compare Blocks.	
	20	Timer / Counters Output Blocks and Registers.	
6	21	Timer / Counters Configuration Examples.	Polling versus Interrupts – Keypad scanning.
	22	Timer / Counters Configuration Examples.	
	23	Analog to Digital and Digital to Analog Conversion Principles.	
	24	The Successive Approximation Register ADC – Intro to ADC12	
7	25	<b>Test 2 – excluding ADCs</b>	Pulse Width Modulated Signal generation using the

	26	ADC – Configuration	Timer / Counter – T/C.
	27	ADC – Configuration	
	28	ADC – Configuration	
8	29	ADC – Code Samples	Analog-to-Digital Conversion using the ADC12.
	30	ADC – Code Samples	
	31	ADC – Code Samples	
	32	ADC – Code Samples	
9	33	Utilizing interrupts to provide an efficient user interface	Two week project: An environment monitoring system.
	34	Utilizing interrupts to provide an efficient user interface	
	35	Utilizing interrupts to minimize overhead associated with data acquisition and control	
	36	Utilizing interrupts to minimize overhead associated with data acquisition and control	
10	37	The connection between C and Assembly Language – providing a processor independent programming language	
	38	The connection between C and Assembly Language – how C statements map on to assembly language statements of a processor	
	39	The software tools that allow the connection between C and Assembly as well as modular programming	
	40	<b>Final Review</b>	

**Table 1 Microcomputer Systems Lecture and Laboratory Topics**

### The “Computer Architecture” course and its lab

The purpose of this course is to introduce students to the architecture and logic implementation of a digital computer system. The course focuses on the major hardware components such as: datapath, control unit, central processing unit, memory subsystem, input / output peripherals and on instruction set architectures. The lab sessions cover the design, simulation and implementation of a 4-bit microprocessor core.

A student who successfully fulfills the course requirements will have demonstrated:

- an ability to list the:
  - building blocks of a computer system
  - types of Programmable Logic Devices (PLD)
  - components of a Central Processing Unit (CPU)
  - types of computer architectures in terms of instruction set and design methodology of the components in the CPU.
- an ability to explain:
  - the concepts of registers, counters, and their design methods
  - the differences between a static RAM and a dynamic RAM
  - the functionalities and design issues of RAMs, PLDs, and FPGAs
  - the concepts of register transfers, datapaths, and control units
  - the concepts of instruction set architectures
  - the operation and design of Central Processing Units (CPU)
- an ability to apply:
  - the Karnaugh Map to a truth table for sequential circuit design
  - the implementation of a logic function using logic components
- an ability to calculate the number of memory blocks and additional logic to construct larger memory blocks
- an ability to determine:
  - the instruction set architecture needed for a given problem/application.

- the suitable architecture (RISC – CISC) for a given problem/application.
- an ability to classify computer systems on the concepts of instruction set architecture and design methodology
- an ability to design registers, counters, datapaths, control units, and CPUs
- an ability to improve the design of a computer system in terms of speed, hardware complexity, and integration.
- an ability to critique a computer system or a CPU in terms of speed, hardware complexity and instruction set architecture.
- an ability to select the optimal CPU architecture for a given problem

### Course History and Revision:

Until fall 2003 this course has not had an associated lab. Student feedback indicated a declining interest in the topics covered, as the lectures progressed. Further inquiry revealed that a hands-on experience would greatly help in understanding the advantages and disadvantages of the different architectural components of a digital computer. As a result, a two hour weekly lab was introduced starting in fall 2003.

Few computer architecture text books suggest project based lab assignments. A review of these<sup>4-12</sup> revealed that most focus only on specific modules. However, the complexity of these modules, and implicitly the time necessary to design them would not leave enough time to integrate them in a complete computer system. In other words such assignments would be limited in number, and would not offer the student the opportunity to experience in linking and interfacing them together. While it is true that accessing a 64-register register file compared to a 4-register register file increases the addressing logic complexity, from a functional point of view the register file serves the same function: temporary data storage within the datapath of the central processing unit. Because we consider the interface and link between individual modules as important to understand as their functionality, we have developed a unique sequence of project based lab assignments. Each subsequent assignment builds on the previous one, ultimately generating/resulting in a complete central processing unit, with memory and a reduced set of digital inputs and outputs, i.e. a rudimentary reduced instruction set computer<sup>13</sup>.

To accommodate this, we have decreased the complexity of the architecture. All students have to adhere to the following general design specifications:

- Register File
  - Clock Synchronous for writing;
  - Two 4-Bit Registers A and B, both with parallel load capability;
  - 1 Data-In-Bus line;
  - 2 Data-Out-Busses if assigned 3-Bus processor;
  - 1 Data-Out-Buss line if assigned 2-Bus processor.
- Function Unit
  - Arithmetic and Logic Unit developed in lab of week 3.
- Memory
  - 8-Bit Addressable Memory, i.e. 8-bit wide Address Bus;
  - 4-Bit Data-In-Bus;
  - 4-Bit Data-Out-Bus;

- 8-Bit Address signal;
- Asynchronous Write signal, MW = 1 to write, 0 to not write, i.e. read;
- 2 separate Memory Blocks / Address Spaces for Instructions and Data, if Memory System architecture is of type Harvard;
- 1 Memory Block / Address Space if Memory System Architecture is of type Von Neumann – see appendix;
- I/O Peripherals Address Space separate from Data Memory Address Space if I/O Peripherals are I/O mapped – see above; the IN and OUT instructions are used to “READ” and “WRITE” from/to the I/O Peripherals;
- I/O Peripherals Address Space located in Data Memory Space if I/O Peripherals are Memory mapped – see above. In this latter case, the instructions READ and WRITE can be used to “READ” and “WRITE” from/to I/O Peripherals, therefore the IN and OUT instructions are not implemented. Instead, PUSH and POP must be implemented – see the STACK paragraph.
- Program Counter – PC
  - One 8-Bit Up Counter with parallel load capability;
  - Clock Enable and RESET Signals;
  - Provides the address value to the instruction memory (segment), that points to the next instruction to be fetched and executed.
- Memory Address Register – MA
  - Two 4-Bit registers, MA-High and MA-Low, together form 8-Bits, which address the Data Memory (Segment).
  - Parallel Load capability;
  - Clock Enable and RESET Signals.
- Input / Output Peripherals
  - There will be 4 I/O Peripherals;
  - The first and second will be Output Peripherals, i.e. two 4-bit registers which will hold the data to be displayed for the two 7-segment display devices on the UP1/2 boards;
  - The third and fourth will be Input Peripherals, i.e. two groups of 4 Tri-State Buffers each; the inputs will be connected to 2x4 switches, the outputs to the internal write-back /destination bus; these are used to read into A the status of the switches to which they are connected;
  - The enables to the output registers and the selects of the input buffers are synthesized using the 8 Address bits/lines, and specific control signals generated by the Control Unit.
- Stack (Only for processors with Memory-Mapped I/O Peripherals)
  - 4-Bit Data
  - 4 Locations deep
  - Push and Pop functionality
  - No Full or Empty check.

In addition, each student follows individual design specifications, which are inferred by decoding a 5-bit binary code according to the following:

Bit	Value = '0'	Value = '1'	Comments
B0	Von Neumann	Harvard	Memory System Architecture
B1	I/O Mapped	Memory Mapped	I/O System Architecture
B2	Hardwired	Micro-Programmed	CU Implementation
B3	3-Bus	2-Bus	Internal Bus Architecture
B4	Tri-State	Multiplexer	2 <sup>nd</sup> & 3 <sup>rd</sup> BUS implementation

**Table 2. Individual Design Specifications**

These translate into 32 different computer designs, all linked together by their common instruction set, shown below.

#	4-Bit Instruction Code *	Instruction Mnemonic	RTN Description	Comments
0	0000	ADD	$C\#A \leftarrow (A+B)$	Four-bit result gets stored in A. Carry out gets stored in C.
1	0001	SUB	$C\#A \leftarrow (A-B)$	
2	0010	INC	$C\#A \leftarrow (A+1)$	
3	0011	DEC	$C\#A \leftarrow (A-1)$	
4	0100	NOT	$A \leftarrow (A')$	
5	0101	AND	$A \leftarrow (A \bullet B)$	
6	0110	OR	$A \leftarrow (A + B)$	
7	0111	XOR	$A \leftarrow (A \oplus B)$	
8	1000	JMPU	See JMPC below for C='0'	Jump Unconditional
9	1001	JMPC	$MAH \leftarrow M[PC] ; PC \leftarrow PC+1 ;$ $MAL \leftarrow M[PC] ; PC \leftarrow PC+1 ;$ $C=0 \rightarrow PC \leftarrow MA$	Jump Conditional  <b>← This is an 8-bit transfer!!!</b>
10	1010	SWAP	$A \leftarrow B, B \leftarrow A$	
11	1011	CPY	$B \leftarrow A$	REG-A remains Unchanged
12	1100	WR	$MAH \leftarrow M[PC] ; PC \leftarrow PC+1 ;$ $MAL \leftarrow M[PC] ; PC \leftarrow PC+1 ;$ $M[MA] \leftarrow A$	Write contents of REG-A to memory location
13	1101	RD	$MAH \leftarrow M[PC] ; PC \leftarrow PC+1 ;$ $MAL \leftarrow M[PC] ; PC \leftarrow PC+1 ;$ $A \leftarrow M[MA]$	Read contents of memory Location into REG-A
14	1110	IN	$MAH \leftarrow M[PC] ; PC \leftarrow PC+1 ;$ $MA<7>=1 \rightarrow A \leftarrow SWH:$ $MA<7>=0 \rightarrow A \leftarrow SWL$	Read Input peripheral into REG-A

		PUSH **	STK[0] ← A: STK[1] ← STK[0]: STK[2] ← STK[1]: STK[3] ← STK[2]	Push A onto the stack
15	1111	OUT	MAH ← M[PC] : PC ← PC+1 ; MA<7>=1 → DISPH ← A MA<7>=0 → DISPL ← A	Write contents of REG-A to Output Peripheral
		POP **	A ← STK[0] STK[0] ← STK[1]: STK[1] ← STK[2]: STK[2] ← STK[3]: STK[3] ← 0;	Pop A off the stack
<b>Legend:</b>				
A<3..0> := A Register (4-Bit)		MA<7..0> := MAH<3..0>#MAL<3..0> Memory Address register (8-Bit, divided into two four-bit halves, MAH and MAL)		
B<3..0> := B Register (4-Bit)		C := Carry Bit (a single-bit register that holds the carry from the most recent arithmetic operation)		
PC<7..0> := Program Counter (8-Bit)		SWH<3..0>: Value on the left-hand (high-order) set of four switches on the board. SWL<3..0>: Value on the right-hand (low-order) set of four switches on the board. DISPH<3..0>: Four-bit register whose value is continuously displayed on the left-hand (high-order) seven-segment display on the board. DISPL<3..0>: Four-bit register whose value is continuously displayed on the right-hand (low-order) seven-segment display on the board.		
IR<3..0> := Instruction Register (4-bit) Holds the opcode of the currently executing instruction.		STK[0..3]<3..0> Four-bit, four-location stack. **		
M[0..255]<3..0> := Memory (256 words of four bits each).				

**Table 3. Instruction Set**

Thus, the size of the processor's internal data bus has been reduced to four bits, the size of the register file to two registers, the size of the memory to 256 locations, and the number of instructions to 16. However, the reduction in size has resulted in a sequence of project based lab assignments which can be completed by a student in the time allotted. Furthermore, by implementing a unique and complete computer, each student is exposed to the implementation of each block and to its function within the system. The work and learning experience culminates in week 10 when the computer is physically emulated in a field programmable gate array, and students demonstrate assembly code programs run on their own computers.

The choice of the field programmable gate array platform for prototyping is arbitrary, as prices between vendors are comparable and all offer a student version of the Computer Aided Design Tools. At Rochester Institute of Technology we are using the Altera UP2 boards and the Quartus Design Environment.

The table below shows how the bottom-top design approach taught in the lectures is then practiced in the lab sessions.

Week	Lecture	Lecture Topic	Lab Topic
1	1	Introduction to the course. Prerequisites.	No Lab during the first week.
	2	Digital Logic Review 2	
	3	Digital Logic Review 3	
2	4	The General Purpose Machine. The User's View. The Machine/Assembly Language Programmer's View.	Introduction to the CAD Tool
	5	Computing Machine/System Block Diagram and Views.	
	6	Classification of Computers and Their Instructions. RISC versus CISC. Machine Characteristics and Performance.	
3	7	Computer Instruction Set Architecture.	Arithmetic and Logic Unit
	8	Addressing Modes..	
	9	Register Transfers Notation Language – RTN. Algorithmic State Machines.	
4	10	Architecture and Logic Design for the 1-Bus Lab-SRC. Data Path – Introduction.	Registers, ROM and RAM Memories
	11	Architecture and Logic Design for the 1-Bus Lab-SRC. Data Path.	
	12	Architecture and Logic Design for the 1-Bus Lab-SRC. Data Path. Discussion of 2- and 3-bus Data Path Architectures.	
5	13	Machine Reset. Machine Exceptions/Interrupts. Hardwired Control Unit for the 1-Bus Architecture Lab-SRC – Introduction.	Processor Data Path – Design
	14	<b>Test 1</b>	
	15	Hardwired Control Unit for the 1-Bus Architecture Lab-SRC.	
6	16	Microprogrammed Control Unit for the 1-Bus Architecture Lab-SRC.	Processor Data Path – Simulation
	17	Microprogrammed Control Unit for the 1-Bus Architecture Lab-SRC – continued.	
	18	Pipelining. Pipelined version of the Lab-SRC Instruction Set.	
7	19	Pipelined version of the Lab-SRC Instruction Set. Pipeline Hazards.	Control Unit – Design
	20	Computer Arithmetic. ALU Design.	
	21	Computer Arithmetic. ALU Design.	
8	22	The Memory Sub-System.	Control Unit – Simulation
	23	<b>Test 2</b>	
	24	The Cache.	
9	25	The Cache.	Complete Processor – Simulation
	26	Virtual Memory.	
	27	Digital to Analog Converters.	
10	28	Analog to Digital Converters.	Complete Processor – Emulation
	29	Computer Networks.	
	30	<b>Final Review</b>	

**Table 4. Computer Architecture Lecture and Lab Topics**

### Evaluation and Grading

All three authors are teaching both courses, and all assign 40% of the final grade to the lab and the remaining 60% to other grade items. The reason the lab is so heavily graded is to ensure that a passing grade is received only if a major portion of the lab has been completed. Furthermore, the individual lab grades are not equal, but as follows: L1=0, L2=1%, L3=2%, L4=2%, L5=5%, L6=5%, L7=5%, L8=5%, L9=5%, L10=10%. Lab of week 10 grade is the highest to reward those outstanding students who complete and demonstrate a complete final assignment. Each completed lab assignment is demonstrated to the lab instructor at the beginning of the next week

lab session. During these demonstrations the lab instructor decides the lab grade also based on a short oral examination in which the students have to defend their design choices.

The other 60% of the grade are divided into homework, quizzes, two mid-term tests and a comprehensive final test. Each of the authors/instructors assigns slightly different percentiles to each item. All of these are written based, with the exception of code based homework which have to be practically demonstrated.

### **Student impact**

There are three major areas of student feedback relative to this approach. From the point of view of enhancing the understanding of the subject matter, this augmented laboratory approach has served to help solidify concepts presented in lecture by allowing the students the opportunity to evaluate and experiment with actual hardware, the operation of which they can verify directly. In terms of access to the equipment necessary to conduct these laboratory experiments and projects, the ability to have immediate access to microcomputer and programmable hardware has allowed an increased flexibility in actually working on assignments and laboratory exercises outside of the lab, so that time spent during scheduled lab sessions can be more focused and efficient in terms of interaction between students, faculty and teaching assistants. One noticeable adjustment that the students face is the concept of managing their time in a diligent manner once they have been given access to the facilities to work on their laboratory assignments in a much less time-constrained fashion. Despite definite statements that indicate that the laboratory assignments are analogous to conventional homework assignments, it usually takes one or two weeks to realize that the assigned activities can not be accomplished within the confines of the assigned laboratory sessions. The students come to realize that additional time is necessary, not only to execute the laboratory assignments, but just as importantly, to plan how they will be accomplished, put together a well considered design of software, in the case of the microcomputer course, or hardware, in the case of the computer architecture course. More importantly, they also begin to realize that the extra investment in time allows them self-regulate their work in increments of time and effort that they find to be optimal. This is particularly important in the way of devising a design (software or hardware), implementing it, debugging it and then fully verifying it.

### **What does this two course sequence achieve?**

In the electrical engineering curriculum at the Rochester Institute of Technology, these two courses are the only ones required with an emphasis on digital and computer systems. They are preceded by an “Introduction to Digital Systems” course, and by an “Introduction to Programming with emphasis on C” course. The challenge was to fit into this two course sequence the following:

- the user’s or programmer’s view of a digital computer system, in particular a microcontroller;
- the computer architect’s view of a digital computer system;
- and the logic designer’s view of a digital computer system.

In the end, in the first course, in addition to the user's or programmer's view, provides introduction into the major components of a CPU and a microcomputer system in general as well as insight into the functional relationship between those components. The ability to manipulate the operation of the microcomputer system in a programmatic fashion is also introduced and provides a concrete relationship between the sequence of events dictated by a program and the actual hardware necessary to execute those operations. For example, the concept of a variable representing an actual storage location as well as the framework necessary to address that location is hard to avoid coming to an understanding of. As such, the first course on microcomputer systems focuses not only a programmer's view of the hardware as some platform to accomplish a given task, but also highlights the importance of the functionality the hardware must provide to implement the program's instructions. The second course then provides a natural next step in explaining how that functionality is actually designed and implemented. In other words, at one level, the first course discusses the fundamental operations necessary to implement to execute the basic operations associated with a program and provides the motivation for understanding how those operations are actually designed and created in terms of a relatively comprehensive digital design. In some sense, it would be reasonable to say that it provides a top-down approach to learning the concepts and techniques associated with microcomputers and the design of digital systems with the ability to execute programmed sequences of operations.

## **Acknowledgements**

The authors want to acknowledge the many students who have provided constructive feedback and helped to refine the sequences of lab assignments. We also want to thank the many Teaching Assistants for their hard work and dedication in the successful use and running of these labs over the past three years.

## **Bibliography**

1. Clements, Alan "68000 Family Assembly Language", PWS Pub. Co.,1994
2. Antonakos, James L. "The 68000 Microprocessor", Fifth Edition (Hardcover) Prentice Hall, 2003
3. "The MSP430 User's Guide", [www.ti.com](http://www.ti.com)
4. Hennessey and Patterson, *Computer Architecture – A Quantitative Approach*, 3<sup>rd</sup> Edition, Morgan Kaufmann 2003.
5. Flynn, Michael, J., *Computer Architecture – Pipelined and Parallel Processor Design*, Jones and Bartlett, 1995.
6. Shen, John, P., and Lipasti, Mikho, H., *Modern Processor Design*, McGraw Hill, 2005.
7. Heuring, Vincent, P., and Jordan, Harry, F., *Computer Systems Design and Architecture*, 2<sup>nd</sup> Edition, Pearson Prentice Hall, 2004.
8. Clements, Alan, *Principles of Computer Hardware*, 4<sup>th</sup> Edition, Oxford, 2006.
9. Hayes, John, P., *Computer Architecture and Organization*, 3<sup>rd</sup> Edition, McGraw Hill, 1998.
10. Mano, Morris, M., and Kime, Charles, R., *Logic and Computer Design Fundamentals*, 3<sup>rd</sup> Edition, Pearson Prentice Hall, 2004.
11. Comer, Douglas, E., *Essentials of Computer Architecture*, Pearson Prentice Hall, 2005.
12. Null, Linda, and Lobur, Julia, *Computer Organization and Architecture*, Jones and Bartlett, 2003.
13. Dorin Patru, Konboye Oyake, Eric Peskin, "A Coherent Sequence of Computer Architecture Laboratory Assignments", Proceedings of the 9<sup>th</sup> International Conference on Engineering Education, San Juan, PR 2006.