Overview of a new book on Computer Systems

Umakishore Ramachandran **William D. Leahy Jr.**

May 26, 2006

1. Rationale for a new book

Most undergraduate institutions teach Computer Architecture and Operating Systems as two separate courses. However, it is well known among academicians and practitioners that there is a symbiotic connection between the systems software and the hardware. The design of instruction sets is influenced by high level language. The OS abstractions (such as process, threads, and page tables) are influenced by the details of the processor and memory hardware. The design of the network protocol stack is influenced by the characteristics of the network interface and the vagaries of the physical network. The list goes on… Unfortunately, most students never see this connection when these two courses are offered as distinct stovepipes. At Georgia Institute of Technology, we followed a similar pattern of offering these two courses separately for a long time at the junior level of the UG program. Several factors came together that forced us to reconsider this format. First of all, the discipline of Computer Science has been expanding and includes topics such as graphics, vision, embedded systems, visualization, and human computer interaction. To allow undergraduate students ample opportunity to explore such emerging topics required a rethinking of the "required" or "core" part of the UG CS curriculum. Second. Georgia Tech switched from a quarter to a semester system. This forced us to take a hard look at the curriculum from the point of view of fitting all the required and elective courses within the total credit hours available for the degree. Third, at Georgia Tech, we have a long-standing tradition of involving undergraduates in research. The entry level for systems research was too high for most undergraduates since by the time they took the architecture and OS courses they were ready to graduate. Given all these factors, we undertook a bold new experiment to offer an integrated architecture-OS semester course "Introduction to Systems and Networks" [1] at the **sophomore level** starting from Fall 1999. This course has been a great success since it makes pedagogical sense to the students seeing the system software and hardware issues presented side by side. Further, introducing the students to systems in the sophomore year allows students to get a deeper exposure to systems through additional elective courses in the junior and senior years, and opens the door for research exposure as undergraduates. It is creative thinking like this and other curricular changes that paved the way for innovations in our UG curriculum and the recent birth of the Threads™ concept for organizing the UG program at Georgia Tech [2].

However, teaching such an integrated course posed a huge challenge in terms of textbooks. There are several textbooks that are excellent for the stovepipe model of the curriculum (such as Patterson and Hennessy for architecture [3], Silberschatz *et al.* [4] and Tanenbaum [5] for OS), but there was none for such an integrated approach. Further, the available textbooks did not match the pedagogical style that we intended for this course which we elaborate in the next section. We developed a comprehensive set of notes and slides for the course and used two standard textbooks (Patterson and Hennessy [3], and Silberschatz, et al. [4]) as background

reference material for the students to supplement the course material. We taught the course in this way since Fall of 1999 to Fall of 2004. Finally in the Spring of 2005, we decided to write a book since (a) the students continually communicated to us a need for a textbook that matched the style and contents of our course, (b) there was no book that met the requirements for such a course, and (c) we got sufficient feedback from several of our colleagues from peer institutions that endorsed the need for such a book. The working title for our new book is: **"Computer Systems: An Integrated Approach to Architecture and Operating Systems,"** to stress the vision behind this project.

2. Overview of the book and pedagogical style

There is an excitement when you talk to high school students about computers. There is a sense of mystery as to what is "inside the box" that makes the computer do such cool things as play video games with cool graphics, play music be it rap or symphony, sending instant messages to friends, and so on. The purpose behind this textbook is to take a journey together to unravel the mystery of what is "inside the box." The book takes the viewpoint that what makes the box interesting is not just the hardware but also how the hardware and software work in tandem to make it all happen. Therefore, the path we take in this book is to look at hardware and software together to see how one helps the other to make the box interesting and useful. We call this approach, "unraveling the box": basically look inside the box and understand how to design the key hardware elements (processor, memory, and peripheral controllers) and the OS abstractions needed to manage all the hardware resources inside a box including processor, memory, I/O and disk, multiple processors, and network.

To get a good understanding of what is going on inside the box we have to get a good handle on both the system software and the hardware architecture. This is the intent of this textbook.

Correspondingly, the book is divided into five parts:

- processor and software concepts related to processor (Chapters 2, 3, 5, 6, and 7)
- memory systems and software concepts related to memory systems (Chapters 8 and 9)
- I/O subsystems and software concepts related to devices and device controllers (Chapters 4, 10, and 11)
- Parallel processors and software issues related to concurrent programming (Chapter 12)
- Network connectivity and software issues related to network protocols (Chapter 13)

The hardware and software issues for each of the above five units is treated concomitantly in this textbook.

Another distinguishing feature of the book is the set of online resources. Due to the fact that this book was born out of teaching the course for the last seven years as a required course for all CS majors (3 offerings in each calendar year), there is a significant collection of online resources.

1. We have PowerPoint slides for all the topics covered in the book making preparation and transition (from other textbooks) easy for potential adopters of the book.
2. There is a significant project component that dovetails each of the five units that we enumerated above. We have detailed project descriptions of several iterations of these projects along with software modules (such as simulators) for specific aspects of the projects.

3. We have problem sets and solution keys for the different units (some of the problems are included at the end of each chapter and others available online).

The pedagogical style taken in the book is one of "discovery" as opposed to "instruction" or "indoctrination." Further, the presentation of a topic is "top down" in the sense that the reader is first exposed to the problem we are trying to solve and then initiated into the solution approach. Take for example memory management (Chapter 8). We first start with the question "what is memory management?" Once the need for memory management is understood, then we start identifying software techniques for memory management and the corresponding hardware support needed. Thus the textbook almost takes a "story telling" approach to presenting concepts that students seem to love. Where appropriate, we have included worked out examples of problems in the different chapters to elucidate a point.

We have used the book (online version) for 5 consecutive semesters since Spring 2005. The feedback from the students has been uniformly positive. Here is a verbatim quote from a student:

> *"I thought the book was a good read. Not due to the fact that it's required and that I'm currently taking the course. Never being exposed to any of these low level ideas ever gave me a fairly good perspective on how well the book was at conveying these complex ideas. I'm reading "Understanding the Linux Kernel" as well right now and after reading the first 10 chapters of the CS2200 book everything started to make sense which is a good sign about the book itself. After reading this book I feel confident in my ability to access architecture design issues on a general basis…"*

3. Intended audience and suggested use of the book

The book is intended as a first course in systems for students, preferably in the sophomore year of the undergraduate program. This is the way we have used it at Georgia Tech for the past seven years as a required course for all CS majors, where students coming into this course have had a pre-requisite course that deals with logic design and C programming (currently taught using the book by Patt and Patel [6]).

Where does such a course fit into the continuum of CS curriculum? Students coming into this course should have a good understanding of data structures, structured programming, and basic logic design. Most CS programs around the country give this exposure to students in the first two or three semesters of the UG program. Thus this course would ideally fit in the second semester of the sophomore year.

How many credit hours should be devoted to this course? To cover the material in this book in one course would require about 45 lecture hours (which usually translates to 3 credit hours in a semester system). In addition, since projects make up a significant part of the learning experience, at least 60-90 hours of unsupervised lab time should be dedicated by a student for this course. Thus a course structured around this textbook may account for 4 credit hours in a semester system or 5 credit hours in a quarter system.

What follows this course? This book (and therefore a course structured around this book) is intended to give a broad exposure to all the elements of a computer system: architecture, operating system, and networking. Thus this course will serve as an entry point for students interested in seriously pursuing deeper systems topics. For example, in our CS curriculum, students specializing in systems go on to take as electives "Advanced Computer Architecture" course using the textbook by Hennessy and Patterson (Computer Architecture: A Quantitative Approach [7]), and "Advanced Operating Systems" course using Tanenbaum's textbook (Modern Operating Systems [8]). For students not specializing in systems this course also gives the necessary and sufficient exposure to "core" systems issues allowing them to pursue other areas of specialization (such as theory, graphics, and AI).

Is it possible to use this textbook in a conventional CS curriculum where the students are taught computer architecture and operating systems as separate courses? While this is not the intended use for the book, it can be used for a computer architecture course by simply dropping the system software and networking chapters (Chapters 7, 11, 13, and parts of Chapter 8 and 12). Perhaps this may be used as a transition strategy while moving towards a curriculum change that presents the hardware and software topics in an integrated fashion. It may be a little harder but not impossible to use this book for a standalone OS course. Many of the system software issues (such as scheduling and memory management) refer to the LC-2200 machine model developed in Chapter 2 (Section 2.18). Coverage of this machine model is important to discuss the OS issues; other than this section, the purely architecture oriented chapters (Chapters 2, 3, 5, 6, and 9) may be dropped for a standalone OS course. However, it should be noted that since the book is meant for an introductory course, certain topics (such as deadlocks) are not covered in the book at all and would need to be supplemented if the book is to be used for a standalone OS course.

4. Comparison to other books in this area

There are excellent textbooks in computer architecture (Hennessy and Patterson [3, 7], Tanenbaum [9], and Stallings [10] to name a few). Similarly, there are excellent textbooks in Operating Systems (such as Silberschatz, et al. [4], and Tanenbaum [5, 8]). Unfortunately, for reasons mentioned in Section 1, none of these textbooks serve the vision behind the proposed textbook, namely, presenting architecture and OS topics in a complementary and integrated manner.

The textbook that comes closest in spirit to our proposed textbook is "Computer Systems: A programmers' perspective," by Bryant and O'Hallaron [11]. This is a truly one of a kind book that is intended to help programmers understand the salient features of the "box" from the point of view of developing correct and performance conscious software. Thus the focus of that book is intentionally on application software and the pitfalls in software development that does not account for system effects. The authors (Bryant and O'Hallaron) cover a number of esoteric topics not usually found in a single textbook (divided into three parts: program structure and execution; running programs on a system; and interaction and communication between programs). As the authors state in the preface, "If you study and learn the concepts in this book,

you will be on your way to becoming the rare 'power programmer' who knows how things work and how to fix them when they break."

On the other hand, our book introduces the fundamental principles of computing systems focusing on the inter-relationship between machine hardware and system software. Starry-eyed sophomores are the intended audience, who want to learn how the computer works and not yet ready to create sophisticated application software. Thus we do not deal with issues relating to creating efficient application software. A course for creating efficient software can be a follow on to an introductory course based on our textbook.

5. Chapter by chapter synopsis

Chapter 1 introduces the basic philosophy of the book using a motivating example of playing a multi-person video game. Using this example, we walk the reader through what happens inside the "box" from the time the user clicks the mouse to the time something blows up on the screen! This journey brings to the forefront the interplay between the system software and the hardware and serves as a springboard to launch into the rest of the book.

Chapter 2 deals with processor architecture. The sections are organized to answer questions such as "how do we design an instruction set?" "Where do we keep the operands?" and "How do high level language abstractions map to machine hardware elements?" We incrementally develop the instruction set and the addressing modes through the chapter, and conclude with a simple instruction-set for a machine model (which we call LC-2200 and used as a baseline in the rest of the book).

Chapter 3 deals with datapath and control. After introducing some key hardware concepts, we quickly launch into the design of the datapath and control unit of a processor. Using LC-2200 instruction set as the focal point, we develop the concepts presented in this chapter.

Chapter 4 deals with the different sources of discontinuities during program execution: interrupts, traps, and exceptions. The hardware support needed in the processor to support such discontinuities is discussed.

Chapter 5 presents the basic metrics for assessing processor performance. This chapter serves as a basis for introducing principles of pipelining in the subsequent chapter. The reader is introduced to the rudiments of pipelined processor design in Chapter 6. The problems in pipelined processor design and the solutions are presented in a deliberate manner so that every step we take in the presentation is well explained and there are no inexplicable surprises.

Now that the reader has been introduced to the design of the processor, Chapter 7 deals with the operating systems issue of managing the processor as a resource. In particular, we deal with topics such as data structures in the OS for managing the processor as a resource and the algorithms for scheduling on the processor.

Chapter 8 deals with the software and architecture aspects of the memory subsystem. We present the basic needs of memory management and build up to the modern-day demand paged virtual memory systems and the associated architectural assists (such as TLBs). The chapter covers OS issues such as page replacement policies, and the interaction between the processor scheduler and the memory manager. Chapter 9 introduces memory hierarchy. Armed with the knowledge of what it takes to keep a pipelined processor busy in the presence of demand paged virtual memory, the positioning of this chapter is intuitive to introduce the idea of caches, how they work, and how their design may interact with the operating system.

Chapter 10 turns to I/O covering topics such as programmed I/O and DMA and builds on the basics of program discontinuities that the reader already learned in Chapter 4. The basics of how to interface peripheral devices to the processor are presented in Chapter 10. Complementing this hardware topic is the issue of writing device drivers in system software. Since the hard-drive is the most important peripheral device in a box, special attention is paid in this chapter on understanding the disk drive and scheduling algorithms that is part of the device driver for a disk. Chapter 11 presents details of designing and implementing file systems as the system software abstraction for persistent hardware storage. Design choices in the OS for implementing a file system on the disk are discussed in this chapter.

Chapter 12 deals with multiprocessors and multithreading. The chapter starts with the question "Why multithreading?" and develops the concept, the operating system issues, and the corresponding hardware support for multithreading in modern processors.

Chapter 13 deals with networking: both the evolution of the hardware for connecting computers together as well as the protocols allowing them to talk to one another. The chapter discusses techniques in the protocol stack of the system software for circumventing network errors.

We finally conclude with Chapter 14 that brings all the inter-related topics together.

References

[1] CS2200: Introduction to Systems and Networks, http://www-static.cc.gatech.edu/classes/AY2006/cs2200_spring/

[2] Threads™ An Undergraduate Educational Program of The College of Computing at Georgia Tech, http://www.cc.gatech.edu/content/view/692/144/

[3] D. A. Patterson and J. L. Hennessy, "Computer Organization & Design: The Hardware/Software Interface," Morgan Kaufmann Publishers.

[4] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating Systems Concepts," John Wiley & Sons.

[5] A. S. Tanenbaum and A. S. Woodhull, "Operating Systems: Design and Implementation," Prentice-Hall.

[6] Y. N. Patt and S. J. Patel, "Introduction to Computing Systems: from bits & gates to C &

beyond," McGraw-Hill.

[7] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers.

[8] A. S. Tanenbaum, "Modern Operating Systems," Prentice-Hall.

[9] A. S. Tanenbaum, "Structured Computer Organization," Prentice-Hall.

[10] W. Stallings, "Computer Organization & Architecture: Designing for Performance," Prentice-Hall.

[11] R. E. Bryant and David O'Hallaron, "Computer Systems: A Programmer's Perspective," Prentice-Hall.