Teaching Computer Organization/Architecture With Limited Resources Using Simulators*

Gregory S. Wolffe
Comp. Sci & Info. Sys.
Grand Valley St. Univ.
Allendale, MI USA
wolffe@csis.gvsu.edu

William Yurcik
Applied C. Sci.
Illinois St. Univ.
Normal, IL USA
wjyurci@ilstu.edu

Hugh Osborne
Sch. Of C. & Math.
U. of Huddersfield
W. Yorkshire U.K.
h.r.osborne@hud.ac.uk

Mark A. Holliday
Math. & Comp. Sci.
W. Carolina Univ.
Cullowhee, NC USA
holliday@cs.wcu.edu

Abstract

As the complexity and variety of computer system hardware increases, its suitability as a pedagogical tool in computer organization/architecture courses diminishes. As a consequence, many instructors are turning to simulators as teaching aids, often using valuable teaching/research time to construct them. Many of these simulators have been made freely available on the Internet, providing a useful and time-saving resource for other instructors. However, finding the right simulator for a particular course or topic can itself be a time-consuming process. The goal of this paper is to provide an easy-to-use survey of free and Internet-accessible computer system simulators as a resource for all instructors of computer organization and computer architecture courses.

1 Introduction

The amount of material comprising the field of computer architecture is continuously expanding; hence, the material can only be partially covered in a typical university course. A common response to this problem is an analog of Moore's Law for educators: instructors have progressively revised their courses to use increasingly higher levels of abstraction [4]. For (generalized) example, students learning how computers operated:

- in the 1950s studied the physics of vacuum tubes
- in the 1960s studied transistor circuits
- in the 1970s studied digital logic gates
- in the 1980s studied integrated circuits
- in the 1990s studied networked computer systems.

While the process of abstraction is a natural progression, students must still be exposed to a certain level of detail.

Coincident with this growth of knowledge in the field is the increasing contention for limited teaching resources. This applies not only to laboratory facilities but also to factors such as expertise, preparation time, and the need to provide resource availability to non-traditional students. Maintaining an up-to-date laboratory infrastructure is an ongoing and expensive process; making it accessible to geographically dispersed and employed students is another problem altogether. Likewise, instructors may have a wide range of backgrounds from novice to expert depending on the topic, but it is a common challenge to remain current.

The most important development addressing both of these trends (ever-expanding material, limited resources) is the advent of the CPU simulator [4]. As students in the 1970s/80s used paper and pencil to design CPU components using Boolean algebra and Karnaugh maps, students today can use a CPU simulator to study computer operation by visualizing the interrelated, simultaneous events that occur during program execution. Interactive CPU simulators enable active learning by allowing students to design their own hypothetical machines, to program, execute and debug system software, and to use simulation to understand the operation of real machines.

Two common problems were identified during a recent survey [3] of instructors of computer organization and architecture courses: (1) instructors are not aware that capable CPU simulators exist and (2) if aware of CPU simulators in general, instructors are not aware of particular simulators that meet their pedagogical needs. The goal of this paper is to provide an easy-to-use survey of *free* and *Internet-accessible* computer system simulators as a resource for all instructors of computer organization and computer architecture courses. We are aware of only two limited prior attempts to provide such a service [15,17]. To our knowledge, this work represents the most comprehensive effort to document CPU simulators across the wide ranges of topic areas and intended audiences.

*Some of the work presented here was supported by the National Science Foundation under grant DUE-9850534.

SIGCSE'02, February 27- March 3, 2002, Covington, Kentucky, USA.

The remainder of this paper is organized as follows: Section 2 provides a context by briefly describing the benefits of CPU simulation. In Section 3 we present the rationale used for grouping the CPU simulators. Sections 4 through 9 describe different categories of CPU simulators. To round out the presentation we include in Section 10 examples of simulators that have been developed for memory subsystem visualization and analysis. We close with a summary and ideas for future work in Section 11.

2 Educational Benefits of Simulation

Recent papers [1,9] have documented the value of using simulation as a tool for teaching computer architecture. As a support tool, simulators are attractive in the following ways: (1) students learn the underlying details of computer operation at multiple levels of abstraction; (2) students have pervasive access to content when and where they want it, significantly increasing availability to non-traditional students utilizing asynchronous learning; (3) state-of-the-art content is available for many, if not all, topics; (4) many simulators are closely linked with textbooks; and (5) little or no infrastructure cost is incurred.

Many CS programs with limited resources are unable to provide dedicated laboratories with examples of computer architecture; the simulators documented here are free and available for a variety of platforms. Internet-accessible simulators allow student experiments ranging from programming historical machines to creating their own new architectures. The interplay of performance tradeoffs and design constraints becomes real when students attempt to simulate a novel computer system. Lastly, creating a computer system simulation in software is a cathartic educational experience similar to building a real computer in hardware - but less expensive, more flexible in allowing students to make mistakes and recover, and more extensible in building additional functionality upon a core design [16].

3 Categorization of Simulation Tools

The focus of this paper is exclusively on free and Internet-accessible computer system simulators that can be easily integrated into computer organization/architecture courses. Early simulators were all text-based. Current computer system simulators have graphical interfaces that make it possible to provide visual metaphors representing the internal operation of a computer.

We divide these free and Internet-accessible computer system simulators into seven different categories summarized in the accompanying tables. Unless otherwise noted all listed URLs are websites (http://). For each specific type of system we give only one representative simulator. Some systems have multiple simulators; the complete list can be found on the maintained webpage whose URL is given at the end of this paper.

The historical machine simulators and digital logic simulators fill distinct educational voids and automate concepts once explored exclusively by hand on paper. The simple / intermediate / advanced categories generally match the introductory, computer organization, and computer architecture courses found in many CS curricula. Multi-processor simulators extend the advanced microarchitecture category by focusing on parallel architectures with multiple CPUs. The memory subsystem category includes simulators that focus on interactions between the CPU and cache memory.

4 Historical Machine Simulators

The study of computer operation can often be enhanced by using examples of machines that either no longer physically exist (other than in a museum), or if they do exist, are too expensive to justify solely for educational purposes [15]. The use of simulation allows an educator to teach concepts on any machine for which a simulator has been constructed. Oftentimes a machine of historical interest is the best example of an architectural concept, or it may have the best available tutorials and documentation.

Using simulation renders obsolescence less of a factor since machines can be virtually recreated and indefinitely maintained. However, in some cases the substitution of a simulator for a real machine may require careful analysis and simulation modeling expertise. For example, it has been found that non-validated simulators of real machines are likely to underestimate performance [6].

TABLE 1: HISTORICAL MACHINE SIMULATORS

Analytical Engine www.fourmilab.ch/babbage/applet.html	Babbage's machine. Web-based Java applet
Apple IIe quark.netfront.net:6502/	Classic 65C02. Unix w/X Windows
Atari ST www.complang.tuwien.ac.at/nino/stonx.html	MC68000 and chipsets. Unix/X, MSDOS, Win 95
Commodore Amiga www.freiburg.linux.de/~uae/	MC680x0 and chipsets. Unix, DOS, Win32, Mac
Commodore 64 (www.uni-mainz.de/~bauec002/FRMain.html)	Popular game system. Multiple platforms
DEC PDP-8 www.cs.uiowa.edu/~jones/pdp8/	First retail computer. Unix w/X Windows
DEC PDP-11 ftp://ftp.update.uu.se/pub/ibmpc/emulators/	Influential CPU family. DOS
EDSAC http://www.dcs.warwick.ac.uk/~edsac/	1st stored-program service Win32, Mac
Enigma (www.ugrad.cs.jhu.edu/~russell/classes/enigma/)	Nazi encryption machine. Web-based Java applet
Sinclair QL (www.geocities.com/SiliconValley/Heights/1296/q-emulator.html	1984 MC68008 machine. Windows NT/95, MacOS
Turing Machine (www.cs.brandeis.edu/~paulq/Turing/TuringAppletMac.html)	Finite state machine. Web-based Java applet

5 Digital Logic Simulators

Modern computers consist of an extremely large number of very simple structures. Whether an instructor chooses to begin with MOS transistors, boolean logic gates or combinational logic circuits, the goal is the same: a bottom-up learning foundation.

The digital logic simulators in this category depict the operation of the following hardware features: basic logic elements with switching theory; circuit analysis (implementation, minimization); timing systems (propagation delays, hazards); flip-flops/latches/registers; logic structures (multiplexers, decoders, comparators, adders); and storage elements (ROM, PROM, RAM).

TABLE 2: DIGITAL LOGIC SIMULATORS

6.111 Digital Simulator (www.mit.edu/people/ eichin/thesis/usrdoc.html)	Set of macros and C libraries that can be used to create a program that simulates a circuit.
Digital Logic Simulator (www.cs.gordon.edu/courses/ module7/logic-sim/example1.html)	Web-based point-click-drag logic gate simulator illustrating the operation of simple circuits.
Digital Workshop (www.cise.ufl.edu/~fishwick/ dig/DigSim.html)	Displays execution of pre-built and customizable logic circuits. Web-based Java applet.
esim Simulator (www.cse.ucsc.edu/~elm/Software/ Esim/index.html)	Advanced tool to build arbitrarily complex digital logic designs using hierarchical design techniques. Unix with Tcl/Tk
Interactive Full-Adder (www.acs.ilstu.edu/faculty/javila/ acs254/fullAdder/FullAdder.html)	Interactive and intuitive demonstration of the implementation of full adder with logic gates. Web-based.
Iowa Logic Simulator www.cs.uiowa.edu/~jones/logicsim/	Simulator based on specification language for digital systems from TTL/gates to large CPUs. Pascal
MIT Digital Logic Simulator web.mit.edu/ara/www/ds.html	Point-and-click simulator with gates, flip-flops, & logic analyzer. Win NT/95/3.1
Multimedia Logic Kits www.softronix.com/logic.html	Visual design system for design and testing of simple circuits. Win32
Simcir – the circuit simulator www.tt.rim.or.jp/~kazz/simcir/	Intuitive point-and-click digital logic gate simulator, Web-based Java applet or executable.

6 Simple Hypothetical Machine Simulators

The authors agree with critics' claims that learning via a simulator is not the same as experience with a real machine – simulators can be even better! As real machines become more complex, they become less suitable for teaching the concepts typically found in introductory computer organization courses. Simple hypothetical machine simulators can serve an important role by giving students access to the internal operation of a system (which is not possible with real CPUs). When properly designed, hypothetical machine simulators excel at illustrating core concepts such as: the von Neumann architecture; the stored program concept; the principle of sequential execution; the intricacies of data representation; the set of essential instructions; the process of instruction translation; the fetch-execute cycle; and the use of registers. In short, hypothetical simulators allow educators to selectively focus attention on important concepts without getting lost in complex machine-dependent details.

Because of the simplicity of the machines they simulate, the tools in this category are particularly well-suited for adaptation to the Web. An example is that most enduring simple hypothetical machine: the Little Man Computer (LMC) that dates back to MIT in the 1960s. A recent paper [16] describes the evolution of five different LMC simulators from their text-based origins to the current graphical and Web-based versions.

TABLE 3: SIMPLE HYPOTHETICAL MACHINE SIMULATORS

CASLE shay.ecn.purdue.edu/~casle/	HTML forms tool. Experiments with registers, instruction latencies and optimization.
CPU Sim www.cs.colby.edu/~djskrien/	Emulator for building at the register-transfer level. MacOS
EasyCPU (www.cteh.ac.il/ departments/ education/cpu.htm)	Animated basic and advanced Intel 80x86 operation.
Little Man Computer www.acs.ilstu.edu/faculty/javila/lmc/	Visualization of LMC paradigm described in [7]. Web-based Java applet
PIPPIN (www.cs.gordon.edu/courses/cs111/ module6/cpu-sim/cpusim.html)	Binary and symbolic mode CPU simulator highlighting data paths. Web-based applet
oisc & urisc (www.pdc.kth.se/~jas /retro/retromuseum.html)	Extreme RISC – a computer with a single instruction.
Simple Computer Emulator Beachstudios.com/sc/	Machine emulator with memory cells and I/O cards. Javascript

7 Intermediate Instruction Set Simulators

The machine simulators described above are designed to use only simple addressing, a limited instruction set, and a very simple memory model. Intermediate simulators, in contrast, tend to include a more realistic set of addressing modes, a more complete instruction set, a more realistic memory hierarchy, and sometimes an interrupt mechanism. As a result of this attention to completeness, much more realistic programming application is possible.

A secondary benefit of using a simulator with a fairly complete instruction set is the opportunity it provides to explore computer science concepts that are important throughout the broader curriculum [5]. Many students understand high-level concepts better after studying the low-level mechanisms upon which they are based. For example, understanding register transfer language indirect addressing clarifies the concept of using pointers in C, studying the cache protocols used to hide memory transfer latencies helps explain the operation of web/browser cache management protocols, and examining CPU architectural support for high-level languages reinforces language and compiler concepts.

TABLE 4: INTERMEDIATE INSTRUCTION SET SIMULATORS

LC2 www.mhhe.com/patt/	Described in [11]. Unix & Windows
Relatively Simple Computer System Simulator www.awl.com/carpinelli	Described in [2], dual-mode control unit. Web-based or downloadable.
SIMHC12 (www.aracnet.com/~tomalmy/68hc12.html)	Simulator for the MC68HC812A4 microcontroller. Java
AMD SimNow! www.x86-64.org/downloads	Intel x86 simulator. GNU/Linux
SPIM (www.cs.wisc.edu/~larus/spim.html)	Used with [12], MIPS simulator with visual GUI. Unix/Linux/DOS/Win
SPIMSAL (www.cs.wisc.edu/~larus/spim.html)	Described in [8]; uses SAL - extended instruction set. Win3.1 & MacOS

8 Advanced Microarchitecture Simulators

The microarchitecture simulators are designed to allow the observation of machine language execution at the microcode level (e.g. data paths, control units). Advanced simulators can be used to investigate the advantages and disadvantages (e.g. efficiency, complexity) of performance-enhancing techniques such as pipelining, branch prediction and instruction-level parallelism. Some of these simulators are microprogrammable, allowing students to experiment with the design of instruction sets.

Most of the simulators included in this category are associated with a textbook and would be appropriate for an advanced course in computer architecture. Note that there is often supplemental material available (e.g. compilers, execution traces, instruction set handbooks).

TABLE 5: ADVANCED MICROARCHITECTURE SIMULATORS

DLX (ftp://max.stanford.edu/pub/max/pub/hennessy-patterson.software)	Bundled with [10], implements the DLX CPU. Unix
DLXview (yara.ecn.purdue.edu/~teamaaa/dlxview)	Interactive graphical extension of DLXsim. Unix
Mic-1 Simulator www.ontko.com/mic1/	Described in [14] (earlier 1988 edition). Java, Unix, Win
Micro Architecture Simulator www.kagi.com/fab/msim.html	Microprogrammed processor similar to that in [14]. MacOS
MipSim (mouse.vlsivie.tuwien.ac.at/lehre/rechnerarchitekturen/download/Simulatoren)	C++ source code for a simulator emulating a pipelined processor based on [10].
SimpleScalar www.simplescalar.org	Described in [13], toolset for instruction-level parallelism & branch prediction. Unix
SuperScalar DLX (www.rs.e-technik.tu-darmstadt.de/TUD/res/dlxdocu/SuperscalarDLX.html)	Pipelined superscalar mixed behavior/RTL model of the DLX processor.
WinDLX ftp://ftp.mkp.com/pub/dlx/	Windows front-end for DLX simulator. DOS 3.3+, Win3.0+

9 Multi-Processor Simulators

Simulators of multiprocessors are significantly different from uniprocessor simulators. One difference is that multiprocessor simulation requires the emulation of features that do not exist in uniprocessors (e.g. shared interconnection networks and shared memory). Another difference is the result of simultaneous execution; a correct simulation must reflect the fact that instructions on different processors are occurring at the same time. A third difference is in the amount of time required to complete a simulation – this is technically challenging since simulation time tends to grow at least proportionally to the number of processors being simulated.

Consequently, developing accurate multiprocessor simulators with good performance is a research area unto itself. The multiprocessor simulators that are useful instructional aids are also ones that are actively used in computer architecture research. Using these simulators can be more complex than using the uniprocessor simulators in the other categories, so they are most appropriate for advanced computer architecture courses. Note that the last link listed for this category is actually a link to a web page listing many of the multiprocessor simulators currently used in research.

TABLE 6: MULTI-PROCESSOR SIMULATORS

ABSS (arithmetic.Stanford.edu/~lemon/abss.html)	SPARC-based multiprocessors
MINT www.cs.rochester/u/veenstra/	MIPS-based shared-memory multiprocessors
Proteus (www.ee.lsu.edu/koppel/proteus.html)	Both shared-memory and message-passing multiprocessors
RSIM rsim.cs.uiuc.edu/rsim/	Shared-memory multiprocessors using processors with instruction-level parallelism
SimOS simos.stanford.edu/introduction.html	MIPS and Alpha-based uni/multiprocessors; OS boot
Wisconsin Simulator Page www.cs.wisc.edu/arch/www/tools.html	Has links to many multiprocessor simulators used for research

10 Memory Subsystem Simulators

The simulators most appropriate for introductory courses tend to be descriptive, illustrating concepts by depicting computer operation. But advanced students are prepared for detailed performance statistics that allow meaningful analysis, evaluation and comparison. An early lesson these students learn is that factors other than the CPU (e.g. cache hit ratio) affect performance. Our final category includes examples of simulators that are used to model and analyze various memory hierarchies and cache configurations.

The simulators listed in the table below encourage experimentation with different cache memory levels, sizes and degrees of associativity. Several of them operate by "executing" memory reference trace tapes and reporting cumulative memory access metrics.

TABLE 7: MEMORY SUBSYSTEM SIMULATORS

Cacheprof www.cacheprof.org	Tool to quantify cache behavior. Linux on x86
Cache Simulator (www.ece.gatech.edu/ research/labs/reveng/cachesim/)	Specify a cache configuration; analyze results. Web-based
CACTI (www.research.compaq.com/ wrl/people/jouppi/CACTI.html)	Model cache access and cycle times for different memories.
Dinero IV www.cs.wisc.edu/~markhill/DineroIV/	Cache hierarchy simulator for memory reference traces. Unix
PRIMA (www.dsi.unimo.it/ staff/st36/imagelab/prima.html)	Trace-driven cache simulator. Unix
Xcache www.prism.uvsq.fr/archi/softs/XCache/	Cache performance profiling. Unix w/X Windows

11 Summary

The availability of free and Internet-accessible simulators provides instructors with a means of effectively and efficiently presenting the expanding field of computer organization/architecture, despite limited resources. In this paper, we have surveyed and characterized simulator teaching tools that can be easily integrated into a wide range of courses. They represent a conceptual breadth and level of interactivity that is difficult to achieve with other teaching techniques.

One idea for future work in this area involves researching the potential for interoperability between simulators. A related idea is identifying individual simulators that can be gracefully extended through the different categories we have identified. These options would aid instructors by broadening the areas in which simulation could be employed and by reducing the learning curve. See the following URL for future contributions and for a maintained set of the links reported in this paper: <http://www.acs.ilstu.edu/faculty/dldoss/yurcik/caale/caalesimulators.html>.

12 Acknowledgments

The authors would like to thank the following members of the ITiCSE 2000 working group for helping to motivate this work: (in alphabetical order) Kevin Boulding/Seattle Pacific Univ., Co-Chair Boots Cassel/Villanova Univ., Jim Davies/Univ. of Oxford, John Impagliazzo/Hofstra Univ., Co-Chair Deepak Kumar/Bryn Mawr Univ. and Murray Pearson/Univ. of Waikato. We would also like to acknowledge intellectual contributions from Cecile Yehezkel/Weizmann Institute and Ed Gehringer/N.C. State (creator of the Computer Architecture Course Database <http://wwwassign.physics.ncsu.edu/comparch/>).

References

[1] Bruschi, S. M. et. al., Simulation as a Tool for Computer Architecture Teaching, *Summer Computer Simulation Conference (SCSC)*, Society for Computer Simulation (SCS), 1999.

[2] Carpinelli, J.D. *Computer Systems Organization & Architecture*, Addison Wesley, 2001.

[3] Cassel, L., Kumar, D. et. al., Distributed Expertise for Teaching Computer Organization & Architecture, *ACM SIGCSE Bulletin*, Vol. 33, No. 2, June 2001, pp. 111-126.

[4] Clements, A. Guest Editor's Introduction: Computer Architecture Education, *IEEE Micro*, Vol. 20. No. 3, May/June 2000, pp. 10-12.

[5] Clements, A. The Undergraduate Curriculum in Computer Architecture, *IEEE Micro*, Vol. 20. No. 3, May/June 2000, pp. 13-22.

[6] Desikan, R., Burger D. and S.W. Keckler. Measuring Experimental Error in Microprocessor Simulation, *Intl. Symp. on Computer Architecture (ISCA)*, 2001.

[7] Englander, I. *The Architecture of Hardware and Systems Software 2nd edition*, Wiley, 2000.

[8] Goodman, J. and K. Miller, *A Programmer's View of Computer Architecture*, Oxford U. Press, 1993.

[9] Grunbacher, H. Teaching Computer Architecture/ Organisation Using Simulators, *IEEE Frontiers in Education Conference (FIE)*, 1998, pp. 1107-1112.

[10] Hennessy, J. and D. Patterson, *Computer Architecture: A Quantitative Approach 2nd edition*, Morgan Kaufmann, 1996.

[11] Patt, Y. and S. Patel. *Introduction to Computing Systems*, McGraw-Hill, 2001.

[12] Patterson, D. and J. Hennessy, *Computer Organization and Design 2nd edition*, Morgan Kaufmann, 1998.

[13] Stallings, W. *Computer Organization and Architecture 5th edition*, Prentice Hall, 2000.

[14] Tanenbaum, A. *Structured Computer Organization 4th edition*, Prentice Hall, 1999.

[15] Yehezkel, C., Yurcik W., and M. Pearson, Teaching Computer Architecture with a Computer-Aided Learning Environment: State of the Art Simulators, *Intl. Conf. on Simulation and Multimedia in Engineering Education (ICSEE)*, Society for Computer Simulation (SCS), 2001

[16] Yurcik, W. and H. Osborne. A Crowd of Little Man Computers: Visual Computer Simulator Teaching Tools, *Winter Simulation Conference (WSC)*, 2001.

[17] Yurcik, W., Wolffe, G. S., and M. A. Holliday. A Survey of Simulators Used in Computer Organization/Architecture Courses, *Summer Computer Simulation Conference (SCSC)*, Society for Computer Simulation (SCS), 2001.