# A Survey of Hands-on Assignments and Projects in Undergraduate Computer Architecture Courses

Xuejun Liang
Department of Computer Science
Jackson State University
Jackson, MS 39217 USA
xuejun.liang@jsums.edu

***Abstract* - Computer Architecture and Organization is an important area of the computer science body of knowledge. How to teach and learn the subjects in this area effectively has been an active research topic. This paper presents results and analyses from a survey of hands-on assignments and projects from 35 undergraduate computer architecture and organization courses which are either required or elective for the BS degree in CS. These surveyed courses are selected from universities listed among the 50 top Engineering Ph.D. granting schools by the US News & World Report 2008 rankings, and their teaching materials are publicly accessible via their course websites.**

## I. INTRODUCTION

Computer Architecture and Organization is an important area in undergraduate computer science programs. According to the Joint Task Force on Computing Curricula of IEEE Computer Society and Association for Computing Machinery [1], the subjects in this area include (1) Digital logic and digital systems, (2) Machine level representation of data, (3) Assembly level machine organization, (4) Memory system organization and architecture, (5) Interfacing and communication, (6) Functional organization, (7) Multiprocessing and alternative architectures, (8) Performance enhancements, and (9) Architecture for networks and distributed systems. They are often taught in either a two-course sequence or a three-course sequence in most of undergraduate computer science programs. Some of these subjects can be taught at both an introductory level and an upper level.

Teaching the subjects in the computer architecture area can be difficult; students sometimes have trouble getting the points of the subjects under the traditional paper-pencil pedagogy. Some topics in computer architecture area are difficult to understand without proper intuitions. Hands-on assignments and projects such as designing, programming, simulating, and implementing a processor architecture would be able to provide such learning intuitions and will certainly help students learning the course subjects and engage their learning interest.

But, there are so many distinct topics in the computer architecture area and computer architectures can be approached in several different levels. Subjects taught in computer architecture and organization courses will vary from institution to institution and between CS and CE majors in the same institution. Therefore, there could be too many choices in selecting hands-on assignments and projects for a computer architecture and organization course.

It is certainly desirable to have an overall picture of these possible hands-on assignments and projects by categorizing them and then getting their distribution over different categories. To this end, the author surveyed hands-on assignments and projects collected from 35 undergraduate computer architecture and organization courses which are either required or elective for the BS degree in CS. These surveyed courses are selected from universities listed among the 50 top Engineering Ph.D. granting schools by the US News & World Report 2008 rankings [2], and their teaching materials are publicly accessible via their course websites.

There have been substantial research works on the computer architecture education. Computer processor simulators are the most useful tools in the computer architecture education and research. There are several lists of processor simulators available from the Internet, for example, the WWW Computer Architecture Page [3]. Most simulators on these lists are for the research purpose. A survey of simulators used in computer architecture and organization courses can be found in [4]. In order to make a simulator easier to use and suitable for the teaching purpose, graphical interfaces to an existing simulator were created in [5]. There are also many cache simulators [6, 7] to allow students to play with cache memory and memory hierarchy. However, the usage of these tools is often limited to a few individual institutions.

This survey will provide readers an overall picture of major hands-on assignments and projects in the undergraduate computer architecture education, their categorization and distribution, as well as languages, tools, and platforms used in these assignments and projects.

## II. ASSIGNMENT CATEGORIES

Hands-on assignments and projects used for the computer architecture and organization education can be categorized based on their contents, programming languages used, and tools used. In this survey, four main categories A, B, C and D are given as shown in Table I. Several subcategories are also listed within each category. There are totally twelve subcategories.

TABLE I
CATEGORIES AND SUBCATEGORIES

| Categories | Subcategories |
|---|---|
| A. Digital Logic Design | 1. Basic Digital Logic Design |
| | 2. Scalar Processor Design |
| | 3. Cache Design |
| | 4. Superscalar Processor Design |
| B. Assembly Language Programming | 5. Basic Assembly Programming |
| | 6. Advanced Assembly Programming |
| C. High-Level Language Programming | 7. Basic High-Level Programming |
| | 8. Processor Simulator |
| | 9. Cache Simulator |
| | 10. Advanced High-Level Programming |
| D. Exploiting Processor Simulators | 11. Using simulators |
| | 12. Modifying simulators |

### A. *Digital Logic Design*

Digital logic design plays an essential role in studying the internal behalves of a computer hardware component. Designing, simulating, and implementing a processor architecture by using the digital logic approach is certainly a valuable hands-on experience in computer architecture area.

Under this category, the first subcategory called basic digital logic design includes the combinational circuits and major components, the sequential circuits and finite state machines, and the pipelined circuits. The knowledge from this subcategory is necessary for students to move forward on the remaining three subcategories. The scalar processor design subcategory contains the single-cycle processor and the pipelined processor. Note that the survey did not include a course whose assignments and projects were solely on the first subcategory.

### B. *Assembly Language Programming*

The ISA (Instruction Set Architecture) level of computer organizations is the interface between software and hardware. A good ISA makes the computer hardware easy to be implemented efficiently and makes the computer software program easy to be translated into an efficient code. Assembly language programming is the best way to study an ISA.

The basic assembly programming subcategory includes (1) System I/O, ALU operations, and control flows, (2) Stacks, subroutines, and recursions, and (3) Programmed I/O, interrupts, and exceptions. One advanced assembly programming example is writing a simple timesharing OS kernel on a processor.

### C. *High-Level Language Programming*

It is true that understanding computer processor and system architectures will help students writing more efficient high-level language programs. But, on the other hand, developing a software program to simulate a computer processor or a computer system will also help students understanding better computer processor and system architectures.

Within the high-level language programming category are listed four subcategories as shown in Table I. The basic high-level programming subcategory deals with using strings, pointers, memory allocation, and etc. The processor simulator subcategory involves in developing and implementing assemblers, functional processor simulators, cycle-accurate processor simulators, and graphical interfaces of processor simulators. Cache simulators are used to simulate the computer cache memory. Two examples in the advance high-level programming subcategory are the parallel programming on clusters with MPI and the development of an interpreter to simulate UNIX file system. Note that the survey did not include a course whose assignments and projects were solely on the basic high-level programming subcategory.

### D. *Exploiting Processor Simulators*

Computer processor simulators are very useful tools in the computer architecture education and research. A functional processor simulator allows users to run an assembly language program and to examine the contents either inside the processor or inside the memory. Graphical computer simulators which are able to visualize dynamic behaviors of a computer will certainly help students learning the computer architecture and engage their learning interest. Some processor simulators allow user to study the processor performance and cost and to compare the processor design tradeoffs before building the processor.

Two subcategories are listed under the exploiting simulator category as shown in Table I. The using simulator subcategory deals with studying dynamic behaviors, performance, and cost of a computer through using simulators. The other subcategory is about modifying an existing processor simulator to add more functions and features."

## III. SURVEY RESULTS AND ANALYSES

The survey was performed for 35 undergraduate computer architecture and organization courses from universities among the 50 top Engineering Ph.D. granting schools by the US News & World Report 2008 rankings. The surveyed courses were taught either during or before the summer of 2007. Among the surveyed 35 courses, 27 courses are required and 8 courses are elective for the Bachelor degree of Science in Computer Science. Because the programming/lab assignments and projects are the focus of this survey, courses without programming/lab assignments or projects, or whose programming/lab assignments and/or projects were not publicly accessible via their course websites, were not considered in this survey.

The survey content includes course descriptions, course formats, student evaluation, textbooks, languages and tools used, and programming/lab assignments and projects. A course is selected in this survey is based on the course description and the availability of programming/lab assignments and projects. A surveyed course may, or may not, be associated with a lab, or may be purely a lab. Among the 35 surveyed courses, 4 courses did not have required textbooks and 3 courses adopted two required textbooks (one for computer architecture and the other for digital logic). The famous textbook "Computer Organization & Design: The Hardware/Software Interface" by David Patterson and John Hennessey [8] was adopted by 17 required courses and 1 elective course. Another textbook

"Computer Architecture: A Quantitative Approach" by John Hennessy and David Patterson [9] was adopted by 5 elective courses.

## A. *Assignment and Project Distribution*

Table II lists the categorization of hands-on assignments and projects under the four categories and the twelve subcategories described in Section 2 for each of the 35 surveyed courses. The column # shows the course number and the column WT shows the weight percentage that assignments and projects will contribute to the final grade of students. The weight of 100% indicates that this is a pure lab course; NA means that the weight is not available. Take an example in Table II, the course # 01 is a required course and its programming/lab assignments and projects will take 35% of the final grade and belong to categories A and B or subcategories 2, 5, and 6. Take another example in Table II, the course # 03 is an elective course and its programming/lab assignments and projects will take 25% of the final grade and belong to category C or subcategory 10.

Note that the number of assignments or projects in each subcategory or category is not recorded in Table II because the workload in each assignment or project may be very different. It is also possible that a big project may be involved in several subcategories or even several categories.

TABLE II
ASSIGNMENT AND PROJECT CATEGORIZATION
(* This is an elective course)

| # | WT (%) | Assignment and Project Categories (Subcategories) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | | | | B | | C | | | | D | |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 01 | 35 | | x | | | x | x | | | | | | |
| 02 | 25 | | x | x | | x | x | | | | | | |
| 03* | 35 | | | | | | | | | | x | | |
| 04 | NA | x | x | | | x | | x | | x | x | x | |
| 05* | 50 | | x | x | | | | | x | | | | |
| 06 | 30 | | | | | x | x | x | | x | | | |
| 07 | 40 | | | | | x | | | x | x | | | |
| 08* | 30 | x | x | x | x | | | | | | | | |
| 09 | 55 | | x | | | x | | | x | | | | |
| 10 | NA | | x | | | | | | x | x | x | | |
| 11 | 50 | | | | | x | | x | | | | | |
| 12 | 100 | | x | | | | | | x | | | | |
| 13 | 20 | | | | | | | | | x | | | |
| 14 | 30 | x | x | | | x | x | | | | | | |
| 15 | 60 | | | | | x | | x | | | x | | |
| 16 | 40 | | x | | | x | | | x | | | | |
| 17 | 25 | | x | | | | | | | | | | |
| 18* | 50 | | | | | | | | | | | x | |
| 19 | 50 | | x | | | | | | | x | | | x |
| 20 | 60 | | x | | | x | | | | | | | |
| 21* | 50 | | | | | | | | | | | x | |
| 22* | 25 | x | x | x | | | | | | | | | |
| 23 | 25 | | x | | | | | | x | | x | | |
| 24* | 30 | | | | | | | | | | | x | x |
| 25* | 10 | | | | | | | | | | x | | |
| 26 | 40 | x | | | | x | | | | | | | |
| 27 | 35 | | | | | x | x | | x | | | | |
| 28 | 30 | x | | | | x | | | | | | | |
| 29 | 15 | | x | x | | | | | | | | | |
| 30 | NA | x | x | | | | | | | | x | | x |
| 31 | 55 | | | | | x | | x | | | x | | |
| 32 | 20 | | x | | | x | | | | | | | |
| 33 | 100 | x | x | | | | | | | | | | |
| 34 | NA | | | | | x | | x | | | | | |
| 35 | NA | | | | | x | | | | | | | |

Based on the assignment and project categorization for each course shown in Table II, the course distributions over the 12 subcategories and over the 4 categories are calculated and shown in Fig. 1 and 2, respectively. Take an example in Fig. 1, two required courses and three elective courses have assignments or projects in Subcategory 3. Similarly, take an example in Fig. 2, three required courses and three elective courses have assignments or projects in Category D. From Fig. 1 and 2, it can be noticed that assignments and projects in the assembly programming (Category B), in the basic high-level programming (Subcategory 7), and in the cache simulator (Subcategory 7) are all offered in a required course. Assignments and projects in the scalar processor design (Subcategory 2) and in the basic assembly programming (Subcategory 5) are most popular, while there are relatively few assignments and projects in the exploiting simulators (Category D).

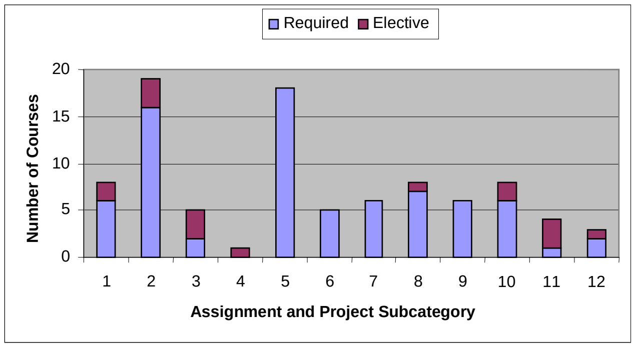

Fig. 1. Course Distribution over the 12 Subcategories

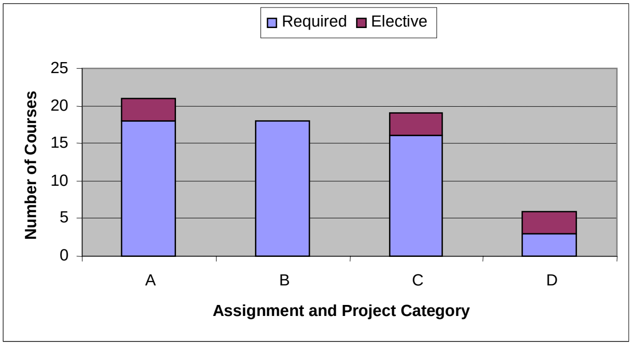

Fig. 2. Course Distribution over the 4 Categories

Based on the assignment and project categorization for each course shown in Table II, the numbers of courses that cover different numbers of subcategories (categories) are calculated and shown in Fig. 3 (Fig. 4). There is only one required course that has assignments and projects in 7 different subcategories as shown in Fig. 3. Aside from this course, any other course covers assignments and projects in less than 5 subcategories.

From Fig. 3 and 4, it can be seen that elective courses tends to focus on less subjects than required courses. Only one elective course covers assignments and projects in two categories; the rest elective courses give their assignments and projects in only one category.

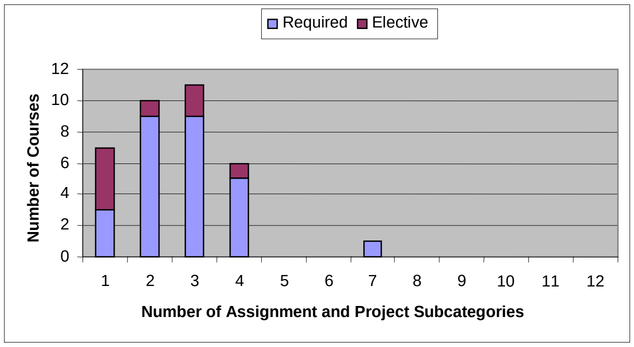

Fig. 3. Course Distribution over Subcategory Coverage

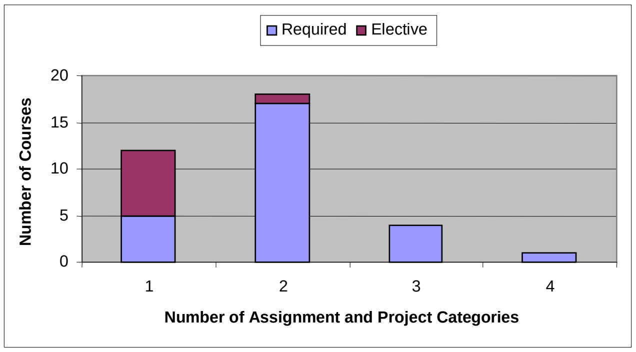

Fig. 4. Course Distribution over Category Coverage

## B. *Digital Logic Design*

In the digital logic design category, the popular design entry is using a hardware description language such as Verilog and VHDL. There are six surveyed courses that use FPGA devices in their assignments and/or projects. A brief overview of FPGA platforms, Logic and FPGA design tools, and processors selected to implement will be given in this section.

There are four FPGA boards used by six surveyed courses. Three of them are commercial products. They are Xilinx's XUP Virtex-II Pro Development System [10], Altera's Development and Education Board [11], and XESS's XSA board [12]. Only one non-commercial FPGA board, called Calinx [13], is from the UC Berkeley.

There are a large number of Logic and FPGA design tools used by the surveyed courses. These tools include the logic synthesis and simulation and the FPGA place & route. Among these tools, many are commercial products, including ModelSim [14], Synopsys VCS [15], Xilinx ISE [16], Altera Quartus II [17], and Aldec Active-HDL [18]. Note that there is generally a free education edition for each of the above commercial tools, but it may be very slow for a large design.

There are also many educational logic design and simulation tools used in the surveyed courses. JSIM is a CAD tool from MIT [19]. It consists of a simple editor for entering a circuit description, simulators to simulate a circuit at device-level, transient analysis-level, and gate-level, and a waveform browser to view the results of a simulation. VIRSIM is a graphical user interface to Synopsys VCS for debugging and viewing waveforms [20]. Logisim is an educational tool for designing and simulating digital logic circuits [21]. Chipmunk system software tools from UC Berkeley perform a wide variety of tasks: electronic circuit simulation and schematic capture, graphics editing, and curve plotting, to name a few [22]. SMOK (pronounced smOk) and CEBOLLITA are tools designed to improve the student design experience in an undergraduate machine organization course [23] in the University of Washington. Funsime/Timsim is a set of Verilog tools used in a surveyed course at Cornell University [24]. Funsim is a functional simulator and Timsim is a timing simulator.

The processors selected to implement in the surveyed courses using the digital logic design is mainly a subset of a well-known processor such as MIPS and Alpha. They may also be a simple artificial processor architecture designed by instructors or even students themselves. Moreover, they can be an educational processor come with a course or a textbook. For examples, Beta is an educational RISC processor used in MIT's Computation Structures course. PAW is a simple architecture designed to be easy to implement in a semester course by the Princeton University. Mic-1 is a microarchitecture described in the textbook "Structured Computer Organization", 4/e, Andrew S. Tanenbaum, Prentice-Hall, 1998. SRC (Simple RISC Computer) is used in the textbook "Computer Systems Design and Architecture," 2/e, Vincent P. Heuring and Harry F. Jordan, Prentice Hall, 2004.

## C. *Assembly Programming*

In the assembly language programming category, the majority of assignments and projects are in the basic assembly programming subcategory. Only five projects are classified in the advanced assembly programming subcategory. They are (1) A simple timesharing OS kernel on the Beta processor at MIT, (2) An interpreter that simulates a subset of the MIPS-I ISA. at Stanford, (3) SPIMbot contest at University of Illinois–Urbana-Champaign, (4) Implement a dynamic memory allocator to study the way structured data types and structures with bit fields supported in MIPS at Texas A&M University, and (5) Create the SnakeOS Operating System on LC-3 at Univ. of Pennsylvania. Note that LC-3 is an ISA used in the textbook "Introduction to Computing Systems: From Bits and Gates to C and Beyond," 2/e, by Yale N. Patt and Sanjay J. Patel, McGraw-Hill, 2003.

The processors that are targeted to the assembly language programming include Beta, MIPS, LC-2K7 (an 8-register, 32-bit computer with 65536 words of memory, designed and used at the university of Michigan–Ann Arbor), PowerPC, IA-32, PAW, LC-3, SRC, and x86. The functional simulator for interpreting, executing, and debugging assembly programs are BSIM for Beta, SPIM [25] and GMIPC [26] for MIPS, LC-3

Simulator [27] and PennSim [28] for LC-3, and SRC Assembler and Simulator [29].

### *D. High-Level Programming*

In the high-level language programming category, the high-level languages used are C, C++, and Java. The processors simulated in the subcategory 8 include MIPS, LC-2K7, a student-designed ISA, PAW, and LC-3. There are eight assignments and/or projects in the advanced high-level language programming subcategory. Five of them are (1) Parallel programming on clusters with MPI at Stanford, (2) An interpreter to simulate UNIX file system at Berkeley, (3) MIPS Multicore Simulator, and Multiplayer Network Tetris Game at Cornell, (4) Use shared memory (pthreads) and message passing (MPI) to compute the *N*th prime number at Duke, and (5) Write a multiprocessor program to do Quicksort running on the MulSim [30] shared-memory multiprocessor simulator at UC-Davis.

### *E. Exploiting Processor Simulators*

It can be seen from Table II, there are only a few courses involved in this category. In the subcategory 11, one project at Berkeley is to determine cache parameters using CAMERA and study virtual memory using CAMERA and VMSIM, where CAMERA is a simple cache simulator used in CS 61C at Berkeley and VMSIM [31] is a simulator of a computer system executing concurrent processes into which desired CPU scheduling and memory management policies can be plugged in with ease. There are also three SimpleScalar [32] assignments in this subcategory to do benchmarking, branch prediction algorithms, a cache memory system, chip multiprocessors, and multithreaded processors. There are three assignments in the subcategory 12: (1) MIC-1 microcode modification, (2) The code modification of sim-outorder in SimpleScalar to explore a micro-architectural issue, and (3) Extend the Mac-1 instruction set by adding a MDN instruction.

## IV. CONCLUSIONS AND FUTURE WORKS

This survey presents an overall picture of major hands-on assignments and projects currently used in the undergraduate computer architecture education at the top universities in USA. This work is intended for helping educators to select and/or create right hands-on assignments and projects as well as tools for their computer architecture and organization courses based on their expected course outcomes. A major future work will be evaluating and comparing some of these hands-on assignments and projects as well as tools. Meanwhile, how to adopt these hands-on assignments and projects as well as tools in computer architecture courses at an underrepresented institution can be an interesting work.

## REFERENCES

[1] The Joint Task Force on Computing Curricula of IEEE Computer Society and Association for Computing Machinery, "Computing Curricula 2001 Computer Science Final Report," 2001

[2] U.S. News & World Report, "America's Best Graduate Schools 2008: Top Engineering Schools," available from: http://grad-schools.usnews.rankingsandreviews.com/usnews/edu/grad/rankings/eng/brief/engrank_brief.php

[3] Luke Yen, Min Xu, Milo Martin, Doug Burger, and Mark Hill, "WWW Computer Architecture Page," available from: http://pages.cs.wisc.edu/~arch/www/

[4] W. Yurcik, G. Wolffe, and M. Holliday, "A Survey of Simulators Used in Computer Organization/Architecture Courses," in the Proceedings of the 2001 Summer Computer Simulation Conference (SCSC 2001), Orlando FL. USA, July 2001

[5] C. Weaver, E. Larson, and T. Austin, "Effective Support of Simulation in Computer Architecture Instruction," in the Proceedings of the Workshop on Computer Architecture Education (WCAE), Anchorage AK USA, May 2002

[6] S. Petit, N. Tomás, J. Sahuquillo, and A. Pont, "An execution-driven simulation tool for teaching cache memories in introductory computer organization courses," in the Proceedings of the Workshop on Computer Architecture Education (WCAE), pp.18-24, Boston MA USA, June 2006.

[7] J. Mendes, L. Coutinho, and C. Martins, "Web Memory Hierarchy Learning and Research Environment," in the Proceedings of the Workshop on Computer Architecture Education (WCAE), pp.25-32, Boston MA USA, June 2006

[8] David Patterson and John Hennessey, "Computer Organization & Design: The Hardware/Software Interface," 3/e, Morgan Kaufmann, 2007

[9] John. Hennessy and David Patterson, "Computer Architecture: A Quantitative Approach," 4/e, Morgan Kaufmann, 2006

[10] Xilinx, "Xilinx XUP Virtex II Pro Development System," available from http://www.xilinx.com/univ/xupv2p.html

[11] Altera, "Altera's Development and Education Board," available from http://www.altera.com/education/univ/materials/boards/unv-de2-board.html

[12] "XSA Board V1.1, V1.2 User Manual," XESS Corporation, 2005

[13] "CALINX - EECS150 FPGA LAB BOARD," University of California, Berkeley, available from http://calinx.eecs.berkeley.edu/

[14] Mentor Graphics, "ModelSim," available at http://www.model.com/

[15] Synopsys, "VCS," available from http://www.synopsys.com/vcs/

[16] Xilinx, "Logic Design," available from http://www.xilinx.com/ise/logic_design_prod/index.htm

[17] Altera, "Quartus II Software," available from http://www.altera.com/products/software/products/quartus2/qts-index.html

[18] Aldec, "Active-HDL Overview," available from http://www.aldec.com/products/active-hdl/

[19] MIT, "JSIM," available from http://6004.lcs.mit.edu/

[20] Tutorial: VCS and VirSim, available from http://users.ece.utexas.edu/~dghosh/vlsi1_lab3/web/lab3set2.html

[21] Logisim, available from http://ozark.hendrix.edu/~burch/logisim/

[22] UC Berkeley, "The Chipmunk System," available from http://www.cs.berkeley.edu/~lazzaro/chipmunk/

[23] SMOK/CEBOLLITA, available from http://www.cs.washington.edu/homes/zahorjan/homepage/Tools/SMOK/index.shtml

[24] Funsime/Timsim, available from http://www.csl.cornell.edu/courses/ece314/projects/ece314p3sp07_files/verilogtools.html

[25] SPIM: A MIPS32 Simulator, available from http://pages.cs.wisc.edu/~larus/spim.html

[26] GMIPC – MIPS Simulator, available from http://www.csl.cornell.edu/courses/ece314/gmipc/gmipc.html

[27] LC-3 Simulator, available from http://highered.mcgraw-hill.com/sites/0072467509/student_view0/lc-3_simulator.html

[28] PennSim Simulator Manual, available from http://www.seas.upenn.edu/~cse240/pennsim/pennsim-manual.html

[29] SRC Assembler and Simulator, available from ftp://schof.colorado.edu/pub/CSDA/Simulators+Models/

[30] MulSim Multiprocessor Simulator, available from http://heather.cs.ucdavis.edu/~matloff/mulsim.html

[31] VMSim - Virtual Memory Management Simulator, available from http://lass.cs.umass.edu/~bhuvan/VMSim/

[32] SimpleScalar, available from http://www.simplescalar.com/