

# A First Year Computer Organization Course on the Web: Make the Magic Disappear

Jeff Penfold and J. Kelly Flanagan  
Computer Science Department  
Brigham Young University  
Provo, UT 84602  
penfold@cs.byu.edu      kelly@cs.byu.edu

## Abstract

This paper describes a freshman-level computer organization course designed to remove the magic surrounding the operation of modern computer systems. Students in the course use a virtual machine with a graphical interface (simulating a processor named TSC) to develop assembly language programming skills. They then use digital design techniques to implement TSC and run their programs on it. We have found the course to be successful in bringing the students full circle, showing how binary instructions and simple gates can lead to a working computer system. A text has been written to integrate the lecture topics and the labs. The text, labs, and software tools have been made available on the Web.

## 1 Introduction

In the past, the Computer Science Department at Brigham Young University required all CS students to take a course in digital logic design. While the course established basic fundamentals of logic design, it did not provide sufficient insight into how logic design relates to computer systems. To remedy this situation, we eliminated the original course requirement and overhauled our freshman-level assembly language programming course. The main goal behind the overhaul was to remove the magic surrounding the internal operation of a computer system. With that in mind, we designed the course to 1) teach digital logic design principles, 2) introduce instruction set architectures and assembly language

programming, and 3) define the basic hardware components of a computer system. We then wanted to reinforce and solidify the concepts taught in class by having the students write and execute assembly language programs on a virtual machine and then design and implement the CPU of the virtual machine via a logic level simulation package. We searched for a text that not only taught the concepts outlined above but provided details on how to apply the concepts through the construction and programming of a CPU. While the idea of building a processor in a computer organization course is not new [2, 4, 7], there did not appear to be a textbook available that met our criteria. Some texts came close [5, 8] but none of the texts provided details on building their respective processors. Frustrated that a satisfactory text could not be found, we decided to write our own text, design our own CPU (we call it TSC), and create the labs detailing how to build and program it. We also had to develop an assembler and virtual machine for TSC. Both programs have graphical user interfaces. We have made the text, labs, and software available on the Web at <http://www.cs.byu.edu/courses/cs143/>.

## 2 The New Course

The online course materials consist of a textbook, five assembly language programming labs, six hardware labs, an assembler, a virtual machine, and associated user manuals. The written course materials constitute about 250 printed pages. The first few lectures of the course introduce computer arithmetic, instruction set architecture, register

transfer languages, and assembly language programming. The students are then prepared to begin working on the assembly language programming labs. These labs instruct the students how to create various programs that they will ultimately execute on their completed versions of TSC. The programming labs will be described in more detail later in this paper. The lecture topics then move to combinational logic and CAD tools. After these lectures the students are ready to begin the hardware labs (described later) which step through the logic level implementation of TSC. The remaining lectures include finite state machines, sequential logic design, basic hardware components, datapaths, state machine control, microprogram control, and various other computer organization topics. All lectures are supplemented by applying the concepts to the construction and operation of TSC. Links between the online text and the lab Web pages further help tie the concepts with the implementation.

The programming labs and the first five hardware labs are disjoint so they can be done concurrently. The sixth hardware lab requires the assembly language programs so it can only be started after the programming labs are completed.

Because the classes are large (between 150 and 300 students per semester), it is not possible to accommodate all of the students questions and concerns during class time. It is also not feasible to operate a closed lab [4]. Instead we use open computer labs and a large group of teaching assistants. One to three assistants are in the labs for most hours of the week to provide any help the students need. During Winter semester 2000, six teaching assistants worked a total of 83 hours a week and there were 266 students enrolled in the course. That works out to approximately 20 minutes per student per week.

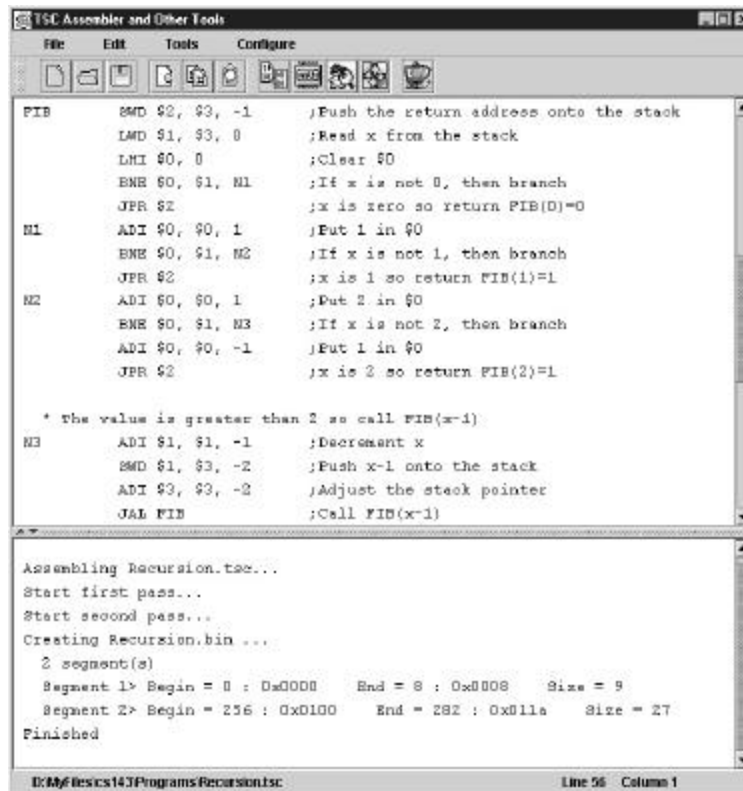
A brief description of TSC and the programming and hardware labs are now in order.

## 2.1 The TSC Processor

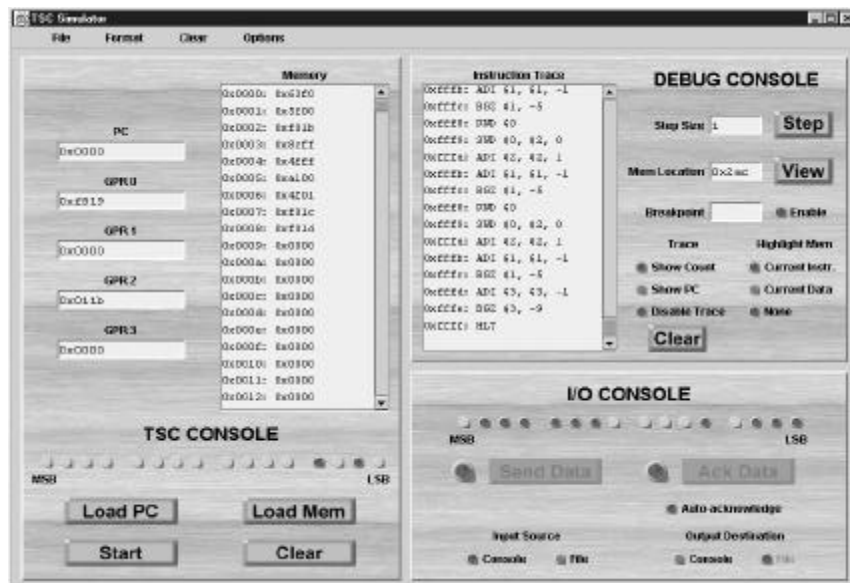
TSC is a 16 bit, load/store, multi-cycle, shared bus processor with a fixed length instruction set. It has four general purpose registers, a 64 kword memory (each word is 16 bits wide), a nine function ALU, and a microprogram control unit. There is one I/O port with dedicated instructions to access it. In the near future, TSC will support memory-mapped I/O. We also plan to convert it to a 32 bit architecture and port the lcc C compiler so that the C-to-machine language relationship can be investigated. Manuals are provided on the Web detailing the programmer's view of TSC. The physical design of TSC is discussed in the text and labs. Ideas for the design of TSC came from various texts that we examined [5, 6, 8]. We kept TSC as simple as possible so that it can be built in one semester.

## 2.2 TSC Assembly Language Programming Labs

While the students are being taught the digital design principles needed to implement TSC in class, they are concurrently working on assembly language programming labs so that they will have working programs to run on their completed processors. In order to facilitate writing the programs, the students are provided with an integrated development environment (IDE) which includes an ASCII text editor, an assembler, and a few other tools. They are also given software which simulates the operation of TSC (i.e., a TSC virtual machine.) This allows them to execute and debug their assembly programs before their circuit design of TSC is completed. We took great care to ensure the graphical interfaces for all the software tools were artistic and easy to comprehend thus reducing the software learning curve. Screen shots of the IDE and simulator are shown in Figures 1 and 2. Note that the TSC simulator interface includes toggle switches and lights giving the user a feel for historical computers. All the software tools are written in Java allowing them to be used on any platform running the Java Virtual Machine. The software is available on the class Web



**Figure 1.** Screen shot of the TSC integrated development environment. It includes an ASCII text editor for writing TSC assembly code, the TSC assembler, and a few other tools.



**Figure 2.** Screen Shot of the TSC simulator. Programs written in TSC assembly language can be executed and debugged on this simulator prior to running the programs on the circuit version of TSC. Programs can be entered into the simulator via the switches on the console or through a file I/O port.

page. User manuals and tutorials for the programs are also provided.

In the programming labs, we not only wanted the students to learn assembly language programming skills, we wanted the programs to deal with real-world computer science applications. This semester the students wrote assembly programs that taught them how to perform basic mathematical operations (compare two numbers, compute absolute values, multiply numbers, etc.), implement and use arrays, construct and traverse a linked list, implement a stack and use it to make procedure calls, etc.

## 2.3 TSC Hardware Labs

As previously stated, our primary means of removing the magic surrounding the internal operation of a computer is to teach the students how to design and build TSC. Instead of using breadboards and electrical equipment, which is expensive and prone to break down, the labs use a CAD package to facilitate the design and implementation of TSC. We chose LogicWorks 3 [1] because it is inexpensive, easy to learn, and provides graphical means for user input and simulation output. The successful use of simulation in computer organization courses is very well documented [2, 3, 4] and does not warrant further discussion here.

A description of each hardware lab follows. The first three labs are completed individually by the students but we allow the last three to be done in groups of two if desired.

### Lab 1: CAD Tools and Logic Gates

The lab starts by encouraging the students to complete a brief tutorial on how to use LogicWorks. They then implement some basic Boolean equations and create a 2:1 multiplexor.

### Lab 2: The ALU

The students start by designing a 1 bit full adder. They then use the adder and their 2:1 MUX to make a single bit of the ALU. Finally they attach 16 ALU

bits together to form the nine function ALU needed for TSC. The nine functions of the ALU are add, subtract, logical AND, logical OR, logical NOT, two's complement, shift right, shift left, and transfer.

### Lab 3: Registers, Memory, etc.

The students are instructed to build flip-flops, registers, large multiplexors, decoders, 8 bit adders, and memories. These are all the basic components needed to construct TSC.

### Lab 4: TSC Datapath

The students are given a description of each element of the datapath (including a description of shared busses and tri-state buffers.) They construct the datapath using the ALU and most of the devices created in Lab 3.

### Lab 5: TSC Control

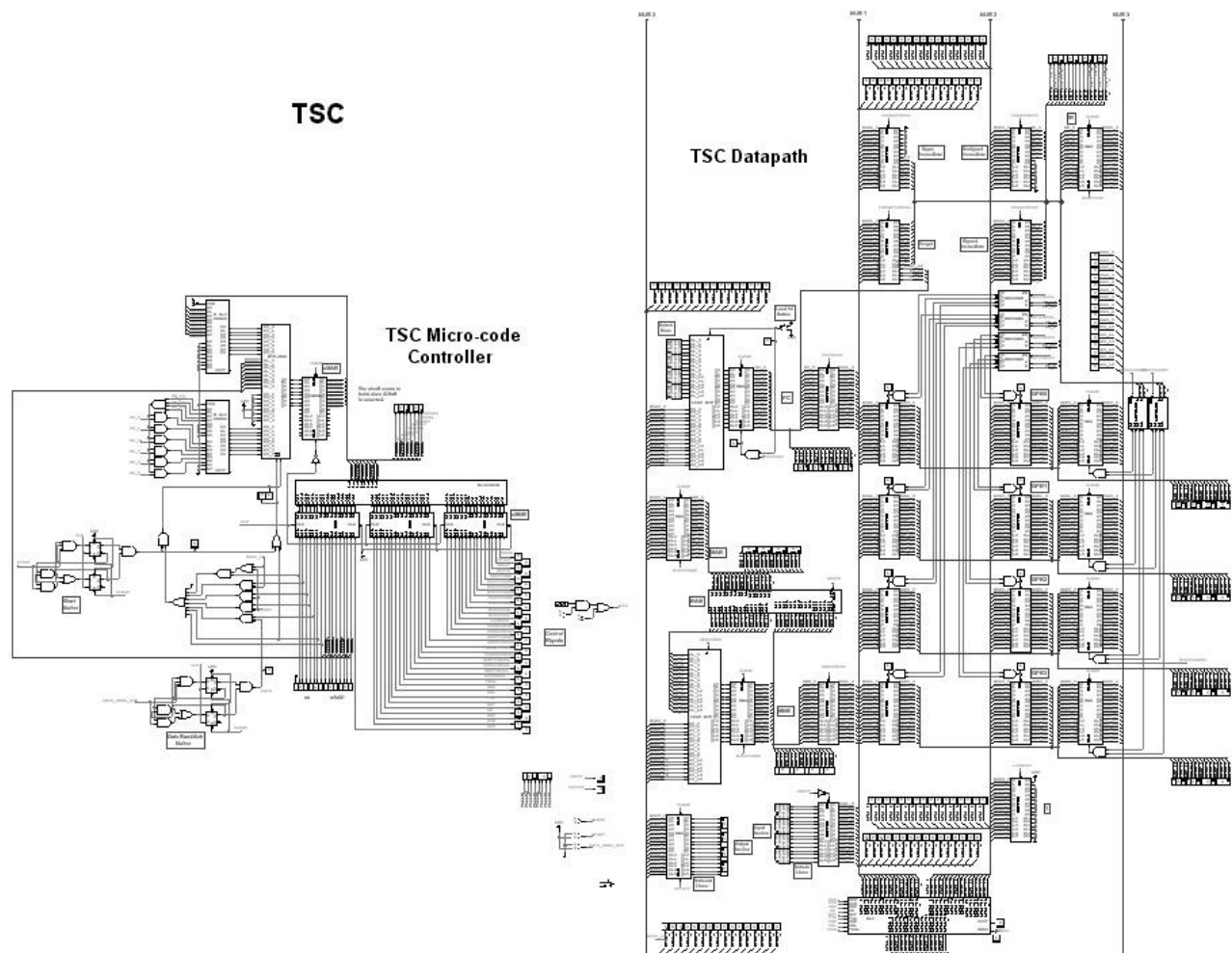
A microprogram control unit is used to control the datapath. (We developed a state machine control unit for TSC but felt it was too hard to debug for a freshman course.) The microcode is supplied so the students only have to build a microsequencer that executes it.

### Lab 6: Complete TSC

The datapath is merged with the control unit, I/O buttons are added, and a TSC binary file is loaded into the memory (they eventually have to load all of the programs they wrote, assembled, and ran on the TSC virtual machine.) If the processor executes all of the students' programs, they have successfully completed the course. Our completed TSC circuitry is shown in Figure 3.

## 3 Results of the Changes

Approximately 800 students have taken the course since we made the changes. Course evaluations have been collected from most of them and we are pleased with the results. As we had hoped, students came out of the course with a greater understanding of logic design, computer organization, and the relationship between them. Students told us that they enjoyed the



**Figure 3.** Our version of the logic level implementation of TSC using the LogicWorks simulation package. Assembled TSC programs can be loaded into the RAM device and executed. The flow of instructions and data through the processor can easily be viewed.

hands-on task of building TSC because it helped them understand, apply, and retain what was being taught in class. One student said that “it was amazing to actually see how the data flows around the insides of a processor. The computer is no longer a mystery.”

Although the labs tended to be time consuming (as expected), almost all of the students were able to complete the labs and demonstrate their working processors. Some of the students told us that the completion of such a large project not only helped them to better understand how a computer processor worked, it gave them the self-confidence to take on large projects in future classes.

We were initially worried about providing help for such a large class but having the open labs staffed by many teaching assistants worked well. The assistants were able to successfully help many students by clarifying difficult concepts and showing them how to debug their projects.

We were surprised by some students who told us that the labs helped them to see the role of software tools in solving large, complex design problems. This lesson was not intended but certainly welcome.

Many of the students mentioned that having the text and labs on the Web made the information they needed readily accessible, both on-campus and at

home. The ability to do some of the lab work at home allowed them to avoid the sometimes crowded open labs and to use their time wisely. Also, having a text integrated with complete lab descriptions often helped them finish assignments without the need of assistance. The dynamic nature of the Web made it easy to update and correct errors in the text and the labs and distribute the corrections to the students quickly.

Putting the course on the Web has benefitted non-BYU students as well. We have received email from people who discovered our Web pages and decided to work through the labs. One such person said “I stumbled across this course [on the Internet] and have been reading the lectures and working the assignments. It is the best site I have ever found on the Internet. I am amazed that anyone would go to the trouble of putting all this information at the disposal of the world. It is very well organized and easy to follow . . . I have ordered Logic Works and will build the computer . . . Thanks for a wonderful course.” Another person told us “I have found your web page to be highly informative and very well written. I used it in class as part of an assignment, and I was pleased to find how easily I could follow the instructions and work the programs.”

## 4 Conclusions

Our first-year computer organization course incorporates digital logic design, assembly language programming, and computer organization complete with an all-encompassing project. The course has proven to be effective in removing the magic that surrounds computers. The TSC project brings the students full circle, showing how binary instructions and simple gates can lead to a working processor. Completing such a large project gives the students a sense of accomplishment and the self-confidence needed to tackle complex projects in future courses. Placing the text, labs, and software tools on the Web enhances the course by allowing the text to be integrated with the labs and user manuals. It also allows students to work on the project at home. The course is an excellent introduction to advanced

computer architecture courses.

The development of this course is ongoing. We welcome all comments and improvements to our text, labs, and library of software tools. In addition, we encourage anyone who would like to use the materials to do so. The course materials are located at <http://www.cs.byu.edu/courses/cs143/>.

## 5 References

- [1] *LogicWorks 3: Interactive Circuit Design Software*, Capilano Computing Systems, Ltd., Addison-Wesley Publishing Company, Inc., 1996. PC and MAC versions available.
- [2] D. Knox, “Integrating design and simulation into a computer architecture course”, Proceedings of the Conference on Integrating Technology into Computer Science Education, ACM Digital Library, June 1997, pp. 42-44.
- [3] D. Magagnosc, “Simulation in Computer Organization: A Goals Based Study”, Selected Papers of the 25<sup>th</sup> Annual SIGCSE Symposium on Computer Science Education, ACM Digital Library, March 1994, pp. 178-182.
- [4] B. C. Parker, “Integrating a Closed Lab Component in Computer Architecture”, Proceedings of the 36th Annual ACM Southeast Regional Conference, ACM Digital Library, April 1998, pp. 185-188.
- [5] Y. N. Patt and S. J. Patel, *Introduction to Computing Systems, From Bits and Gates to C and Beyond*, McGraw-Hill Higher Education, 2000.
- [6] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, The Hardware/Software Interface*, 2<sup>nd</sup> ed., Morgan Kaufmann Publishers, Inc., 1998.
- [7] R. A. Pilgrim, “Design and Construction of VSC”, SIGCSE Bulletin, Vol. 25, No. 1, March 1993, pp. 151-154.
- [8] S. G. Shiva, *Computer Design and Architecture*, 2<sup>nd</sup> ed., HarperCollins Publishers, Inc., 1991.