# Using Simulation In IS Curriculum 2001

Robert F. Zant
Department of Applied Computer Science
Illinois State University
Normal, IL 61790

## Abstract

The simulation of a computer is not commonly used in information systems curricula. However, the use of simulation is popular among computer science faculty. Simulation allows the visualization of the internal workings of a computer and a "hands-on" interface at the machine level. This paper describes the simulation of a simple computer architecture that is appropriate for use in IS'01.1 – *Fundamentals of Information Systems*, IS'01.4 – *Information Technology Hardware and System Software*, and IS'01.5 - *Programming, Data, File, and Object Structures.*

**Keywords:** Computer Architecture, Interpreters, Simulation, IS 2001

### 1. INTRODUCTION

Computer simulation is commonly used in Computer Science curricula to teach fundamentals of computer architecture. The simulation of a computer is much less commonly used in Information Systems curricula. This paper describes the simulation of a simple computer architecture that is appropriate for use in IS'01.1 – *Fundamentals of Information Systems*, IS'01.4 – *Information Technology Hardware and System Software*, or IS'01.5 - *Programming, Data, File, and Object Structures* (Longenecker 2001).

Parker and Drexel (Parker 1996) reported having students use "breadboard" techniques to build a real, though very basic, computer that was then used to learn computer architecture. Others have used simulation packages, such as EasyCPU (HAIT 2002), LMC (Vila 2000), CPU Simulator for Windows (SPA 1997), and SPIM (Larus 2002), instead of real computers in courses on computer architecture, assembly language, and compiler design (Bergmann 1993; Coe 1996; Coey 1993; Yehezkel 2001). Like the direct use of hardware, computer simulations allow "hands-on" interactive learning (Bergmann 1993) and act to motivate students (Yehezkel 2001). However, the use of simulation software has many advantages over using actual computers. Simulation software is less expensive, easily modified, and can be accessed broadly, especially if it is available via the World Wide Web. A simulated computer can also be designed to be as simple or as complex as needed to meet the objectives of the course in which it is used. Simulation provides a mechanism for the effective visualization of computer concepts by allowing the student to "see" and directly interact with CPU registers and memory contents (Barnett 1995).

Unfortunately, the computer simulations used in computer science courses tend to contain features that are not appropriate for use in information systems courses. For example, HASE simulates not one, but several different architectures which include features such as parallel arithmetic units and multiple coherent caches (Coe 1996). Both Searles (Searles 1993) and Bergmann (Bergmann 1993) describe simulators that utilize a micro-code architecture. Barnett (Barnett 1995) describes a relatively simple simulator that has eight instructions and does not include an ALU.

### 2. THE TRIPLE S COMPUTER

The Triple S Computer (Super Simple Simulated Computer) implements the classic von Neumann computer architecture. The simulator has limited features designed to demonstrate basic architecture concepts (see Figure 1).

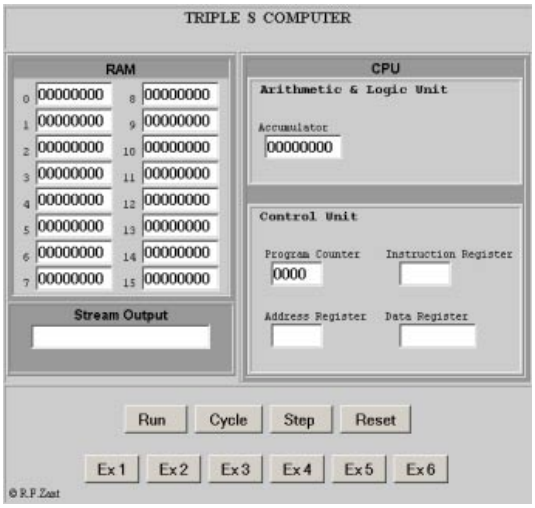


**Figure 1. The Triple S Computer**

The CPU (see Figure 2) contains an Arithmetic and Logic Unit with a single eight bit register, the accumulator, used in arithmetic operations. The CPU also includes a Control Unit with a Program Counter Register (4 bits), an Instruction Register (4 bits), a Data Address Register (4 bits), and a Data Register (8 bits).

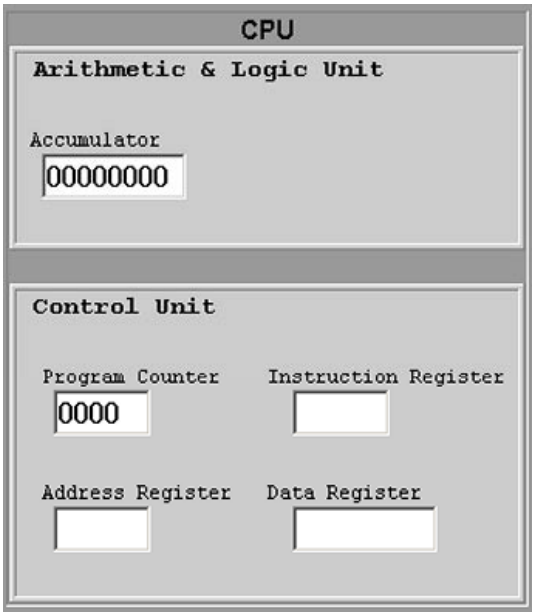


**Figure 2. CPU**

The main memory (see Figure 3) is composed of 16 locations (8 bits each). There is no input component. Instructions and data are entered directly into RAM.

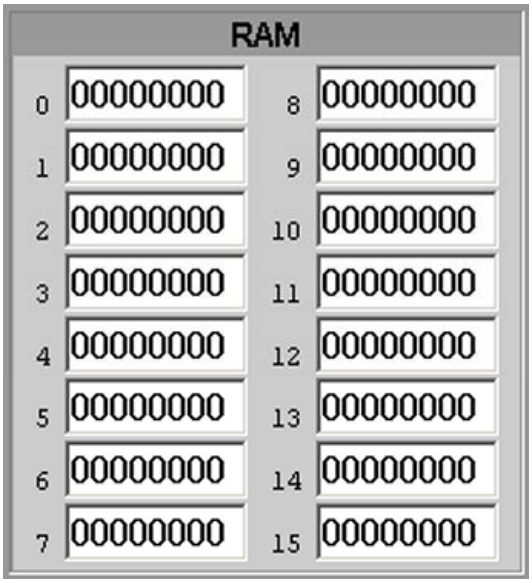


**Figure 3. Main Memory**

Output is displayed in a twenty-four-byte window (see Figure 4). The contents of any RAM location can be displayed as an ASCII character, a binary string, or as a decimal value. Output streams longer than 24 bytes are scrolled.


**Figure 4. Output Window**

This architecture demonstrates a number of basic characteristics of computer organization. The memory helps students visualize the sequential organization of bytes in RAM and the difference between the address of a memory location and its contents. The ALU demonstrates that data must be moved into the CPU in order to operate on it. The Control Unit registers provide a visual trace of the fetch-execute cycle and the relationship between RAM and the CPU. The output window demonstrates the concept of stream output.

**Instruction Set**

An important function of the Triple S Computer is to demonstrate the concept of an instruction and of an instruction set. The instruction set is composed of twelve instructions. The architecture uses a single operand instruction. The operand is either an immediate fixed binary value or an address in RAM.

The instruction set includes four arithmetic operations. One add and one subtract instruction each operate on the contents of a RAM location and the contents of the accumulator. Another version of the add and the subtract instructions operate on an immediate operand and the contents of the accumulator. There are three output operations--one each to display the decimal, the character, and the binary representation of the contents of a RAM location. There are two operations to move data to (Load) and from (Store) the accumulator. There are two jump instructions. One transfers control to the location specified by the operand if the value in the accumulator is zero. The other transfers control if the value is not zero. The final instruction is used to halt execution.

All of the instructions are represented by binary codes. Mnemonic representations of instructions are not used. This stresses that all CPU operations are carried out in binary and provides a background for students to better appreciate the role of compilers and interpreters.

**Data Representation**

Two types of data are defined--integers and characters. All integers are stored as unsigned, fixed binary values. Immediate operands are four bit numeric values. Numeric values stored in RAM locations are eight bits in length. A numeric value can be displayed in the output stream as either a binary value or as its decimal equivalent.

Characters are stored as ASCII characters. The Triple S Computer includes an instruction to display a character stored in any RAM location. There are no other instructions to operate directly on ASCII data; e.g., string operations. Thus character manipulation is very limited. An equal compare of two characters can be accomplished by subtracting one from the other and then either using the jump on zero or the jump on non-zero.

**Operation**

The Triple S Computer is implemented in JavaScript and is available on the World Wide Web (http://www.acs.ilstu.edu/classnotes/acs160/sssc/). Students type both instructions and data directly into the RAM locations. Then the address of the first instruction to be executed is entered directly into the Program Counter register. All values are entered in binary format.

Alternatively, one of six example programs may be selected by clicking on the corresponding button (see Figure 5). Since programs are short (there are only 16 RAM locations), there is no provision for storing or retrieving programs on disk.

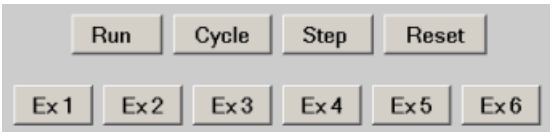


**Figure 5. Operating Buttons**

To execute a program contained in RAM, the student uses the "RUN," "CYCLE," or "STEP" button (see Figure 5). The "RUN" button causes execution to begin and continue until a HALT instruction is encountered or an error occurs. The "CYCLE" button causes one instruction to be executed and then the Program Counter to be incremented. The "STEP" button is clicked three times to execute one instruction. On the first click, the instruction pointed to by the Program Counter is fetched and the Instruction and Address registers are updated. On the second click, the operand is interpreted, the Data Register is updated, and the operation specified by the instruction is carried out. The third click causes the Program Counter to be incremented in preparation for the next instruction.

**Example Programs**

Students can begin using the simulation right away without writing a computer program. Six example programs are included that demonstrate the use of eleven of the twelve instructions. If desired, an instructor can easily change these programs since the system is written in JavaScript.

The first example demonstrates the use of the three types of display and the arithmetic instructions that reference a RAM location. Example number two demonstrates the operation of the add and subtract immediate instructions. The third example shows multiplication by repeated addition. This example demonstrates the programming of a loop using a counter and a jump instruction. The forth example modifies an ASCII character by adding one to its value. This demonstrates the ubiquities of binary representations. (Subtracting 32 from a lower case letter would show how to capitalize a letter). Examples five and six demonstrate error conditions. Numeric overflow is demonstrated in example five and an invalid instruction error is demonstrated in example 6.

## 3. PEDAGOGICAL USE

The Triple S Computer has been used in an introductory course taken by students who are majors or minors in information systems. As described by Barnett (Barnett 1995) simulators can be used in several different ways. The Triple S Computer is used in the classroom to give live demonstrations of the operation of a computer. Outside of class, students use the simulator to complete assignments given in a lab or as homework. Also, students are encouraged to use the simulator in an exploratory mode. Students can modify examples given in class or make up their own problems.

The Triple S Computer is also used as an example of an interpreter in the section of the course dealing with compliers and interpreters. Advanced students are encouraged to study the JavaScript source for the interpreter and even make their own enhancements.

Among the concepts that can be covered using the Triple S Computer are:

**Basic Computer Architecture**

The Triple S Computer contains RAM with sixteen bytes and a CPU with five registers. The memory helps students visualize the sequential organization of bytes in RAM and the difference between the address of a memory location and its contents. The accumulator demonstrates that instructions usually operate on data contained in CPU registers. The Program Counter, and the Instruction, Data, and Data Address registers provide a visual trace of the fetch-execute cycle and demonstrate the relationship between RAM and the CPU.

**Machine Level Instructions**

The Triple S Computer demonstrates that a computer instruction specifies a basic operation that the CPU is designed to perform and, typically, an operand. All instructions except HALT include either an immediate operand or a single address operand. Triple S Computer instructions are all of fixed length. Depending on the depth to be covered in a course, this architecture can be used as a springboard to discussing multiple-operand instructions and variable length instructions.

**Instruction Set**

A CPU is designed to execute certain predefined instructions. The Triple S Computer's instruction set contains twelve different instructions. Even with the limited set, the concept of constructing complex operations from the available instructions can be demonstrated. For example, multiplication through multiple additions is easily demonstrated.

**Stored Program**

Students enter instructions and data directly into RAM locations. The concepts of stored programs and memory allocation are explicitly demonstrated by the student selecting where to enter instructions and data values in RAM.

**Sequential Execution (Program Counter)**

The "CYCLE" button is used to execute a single instruction. The Program Counter and the Instruction register allow students to trace the normal sequence of execution of a program and to "see" how a jump instruction modifies the Program Counter.

**Fetch/Execute Cycle**

A unique feature of the Triple S Computer is its ability to demonstrate the detail of the instruction execution cycle. Students use the "STEP" button to break the instruction execution cycle into three steps—instruction fetch, instruction execution, and the incrementing of the Program Counter. The process is visually followed as the CPU registers are updated at each step.

**Binary Numbers and Arithmetic**

Students enter binary numbers as immediate operands and as fixed binary values in RAM. Binary addition and subtraction can be programmed with the results observed directly in the accumulator and displayed in the output window as either a binary string or as the decimal equivalent.

**Binary Codes**

Students are introduced to binary codes both as instruction codes and as ASCII coded characters. The fact that a byte may be interpreted in various ways can be demonstrated, for example, by capitalizing an ASCII alphabetic character by subtracting 32 from its code.

**Output**

Three instructions provide the capability to display the content of a RAM location. One displays the content as an ASCII character, another as a decimal value, and the third as a binary value. Output is displayed as a stream. The display instructions can be used to demonstrate that a binary string may be interpreted as either a numeric value or as a character.

**Compilers and Interpreters**

As mention earlier, the Triple S Computer is an excellent platform to use in discussing the differences between a compiler and an interpreter. Once students have used the simulator and entered instructions and data in binary, they have a concrete frame of reference to understand the functioning of a compiler in translating source code into machine code. In fact, small paper-and-pencil problems can be used where the student complies assignment statements into Triple S machine code.

The Triple S Computer provides a relatively simple, working example of an interpreter. Students can study the JavaScript source code and even add new instructions, such as, multiply and divide. Thus the simulator allows students to not just read about the concept of an interpreter but to get hands-on experience programming an interpreter.

## 4. SUMMARY

Simulation has proven to be a very valuable tool in assisting students in understanding basic computer concepts. A simulation provides a hands-on experience that is superior to the operation of an actual computer since students can see the state changes in the registers of the CPU and in memory contents. A simulation of simple computer architecture, the Triple S Computer, was presented. It was designed to be appropriate for use in any of three courses in IS Curriculum 2001; IS'01.1 – *Fundamentals of Information Systems*, IS'01.4 – *Information Technology Hardware and System Software*, and IS'01.5 - *Programming, Data, File, and Object Structures.* The simulator is unique in that it subdivides the fetch/execute cycle into three steps that the student can execute in sequence and it is widely available on the web.

## 5. REFERENCES


Barnett, B. Lewis III, 1995, "A Visual Simulator for a Simple Machine and Assembly Language," ACM SIGCSE Bulletin, 27, pp. 233-237.

Bergmann, Seth D., 1993, "Simulating and Compiling a Hypothetical Microprogrammed Architecture with Projects for Computer Architecture and Compiler Design," ACM SIGCSE Bulletin, 25 No. 2 (June), pp. 38-42.

Coe, P.S., L. M. Williams and R. N. Ibbett, 1996, "An Interactive Environment for the Teaching of Computer Architecture," ACM ITiCSE, Barcelona, June.

Coey, W.A., 1993, "An Interactive Tutorial System for MC68000 Assembly Language Using Hypercard," ACM SIGCSE Bulletin, 25 No. 2 (June), pp. 19-23.

HAIT, EasyCPU, 2002, http://www.hait.ac.il/departments/education/CPU.htm

Larus, James, 2002, SPIM, http://www.cs.wisc.edu/~larus/spim.html

Longenecker, Jr., Herbert E., Gordon B. Davis, David L. Feinstein, John T. Gorgone, and Joseph S. Valacich, 2001, "Undergraduate IS Curriculum 2001," AIS, August, http://www.is2000.org/

Maj, S. P., D. Veal and P. Charlesworth, 2000, "Is Computer Technology Taught Upside Down?" ACM ITiCSE 2000, Helsinki, Finland, pp. 140-143.

Maj, S. P., D. Veal and R. Duley, 2001, "A Proposed New High Level Abstraction for Computer Technology," ACM SIGCSE 2001, pp. 199-203.

Parker, Brenda C., and Peter G. Drexel, 1996, "A System-Based Sequence of Closed Labs for Computer Systems Organization," ACM SIGCSE Bulletin, 28, pp. 53-57.

Reid, R. J., 1992, "A Laboratory for Building Computers," ACM SIGCSE Bulletin, 24, pp. 192-196.

Searles, Delmar E., 1993, "An Integrated Hardware Simulator," ACM SIGCSE Bulletin, 25 No. 2 (June), pp. 24-28.

SPA, 1997, CPU Simulator for Windows, http://www.spasoft.co.uk/

Vila, Joaquin, 2000, LMC, http://www.acs.ilstu.edu/faculty/javila/lmc/

Yehezkel, Cecile, W. Yurcik, and M. Pearson, 2001, "Teaching Computer Architecture with a Computer-Aided Learning Environment: State-of-the-Art Simulators," International Conference on Simulation and Multimedia in Engineering Education (ICSEE), Phoenix, January.