Development of a Java2-based Tutorial Binary Calculator for the Instruction of Binary Arithmetic

Gerard N. Foster
Purdue University, School of Technology, Kokomo, Indiana

Abstract

This paper describes instructional software developed to teach binary arithmetic. The heart of the software is a binary calculator written in Java2. This tutorial calculator performs addition, subtraction, and the logical operations, AND, OR, XOR and XNOR. There are two modes of number entry. The problem values are either generated randomly or are entered by the student. A log of the number of attempts, the number good and the percentage good is displayed for problems worked in the random-entry mode. Right and wrong bits in the student's answer are highlighted in green and in red, respectively. Attributes of the calculator are presented. Methods of software development are explained and a brief account of use with students is given. Because the calculator is written in Java, new objects can be created to extend the capabilities while keeping the old functionality. Also it can be integrated into other instructional software on the web. The preliminary work on this software was funded by MIDC (Multimedia Instructional Development Center) at Purdue University.

Background

Binary numbers and binary arithmetic emerge at various points in the technology curriculum, often with increased complexity. As a student progresses through the curriculum, the binary numbers they encounter increase in size and the binary formats (signed, unsigned, fractional, integer, etc.) vary according to the application. At the lower levels, the students are not ready for examples and laboratory applications that illustrate and cement into memory the concepts of binary operations and formats required throughout their academic career. Thus, as educators, we must revisit and extend the coverage of binary numbers, often with time constraints imposed by the need to cover technical applications and circuits of the moment. The list below is a summary of number concepts that can be generally illustrated at that level. Higher concepts may be introduced but the applications are not readily introduced as laboratory exercises.

Fundamental digital courses (freshman year)
- Number conversion (Binary, decimal, hexadecimal, BCD)
- Logical operations (AND, OR, NAND, NOR, XOR, XNOR)
- Arithmetic operations (Add, Subtract)
- Signed and unsigned numbers.
- Number size generally 1 to 4 or 8 bits.

Microprocessor/microcontroller courses (sophomore year)
- Fractional numbers for scaling of A/D conversions.
- Scaling, offsetting and conversion to BCD.
- BCD arithmetic for clocks.
- Carry, half carry, overflow, negative, and zero flags.

Bit setting and clearing.
Number size generally from 8 to 16 bits.

Digital signal processors and advanced microcontroller subsystems (junior year)
Fractional and mixed number formats.
Fixed and floating point considerations.
Signed Multiplication (numbers can be mixture of signed and unsigned numbers).
Overflow protection with saturated mode of calculations.
Number size generally from 16 to 32 bits (40 bits with overflow).

It is with this broad need to teach the principles of binary number manipulation across the curriculum that this work evolved. The primary goal was to provide instructional material to teach binary concepts. The secondary goal was to use state-of-the-art software, the knowledge of which could be transferred to creating instructional tools in other areas.

The calculator

The author decided that the instructional tools should be self-standing, interactive and should include both text and graphics. Java, being an object-oriented language with graphical capabilities suitable to creating software that can be distributed via Internet, was selected as the development tool for the interactive portion of the instructional software. After writing some traditional textual material, the need to illustrate the points with interactive software soon emerged and the road to developing a Java-based tutorial calculator began. The graphical interface of the calculator as it exists today is presented in the following figures.

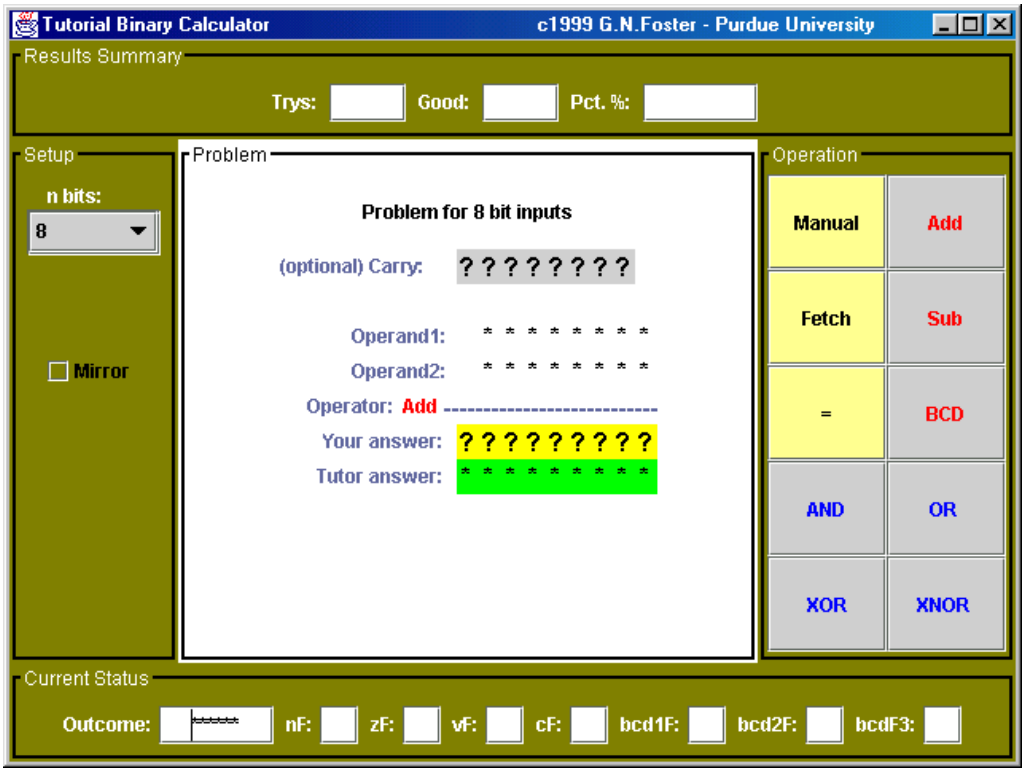

Figure 1: Binary calculator at start-up.

At start-up, the calculator presents a central problem panel surrounded by panels organized into functional groups. These panels and their functions are summarized below.

Problem panel

Problem values are presented. Operand1 and Operand2, the problem source values, are selected with either the Manual button or the Fetch button (see Operation panel).

The student enters a bit in the 'Your answer' field by click the mouse button over the appropriate bit. The background of these bits is yellow when the student is to enter values in these locations.

The 'Tutor answer' field has a green background. Its values are filled by the calculator software after the '=' sign in the operation panel is pressed.

The Carry field is an optional entry field that has a gray background.

Operation panel

The 'Fetch' button determines and displays a set of random binary numbers in the Operand1 and Operand2 fields. It also causes the Results Summary panel's 'Trys' display to increment and the Current Status panel's 'Outcome' display to display the word 'Process.'

The 'Manual' button allows the student to enter problem values into operands 1 and 2.

The '=' button performs the calculation. The tutor answer is displayed. The bits in the student's answer are highlighted individually in green or red depending on whether they are right or are wrong. The Results panel's displays are updated as are the displays for the Current Status panel.

The remaining buttons determine the type of operation that the calculator will perform. They also set up the problem display. For example, the 'Add' and 'Sub' operations require a carry bit whereas the logical operations, such as 'XOR' (exclusive-OR), do not. The logical operations are performed as a bit-wise calculation similar to the way a microprocessor would do them.

Results panel

The Results panel displays the number of attempts, the number good, and the percentage good. These results are only updated if the original problem numbers were obtained randomly by pressing the Fetch button. The number of attempts is incremented before the problem is completed to prevent the student from switching to a different problem when presented with a challenging set of numbers.

Current status panel

The Current status panel presents the outcome as 'Correct' with green background, 'Wrong' with red background or 'Process' with yellow background. The flags indicate the outcomes of the prior calculation. A carry or borrow operation in an addition or subtraction causes the cf flag to become a '1.' The result of zero causes the zero flag to be '1.' If the most significant bit before the carry bit is a '1' the negative flag becomes '1." The overflow flag acts in the same way as that of a standard microprocessor. If the addition of two positive numbers

results in a negative number or the addition of two negative numbers results in a positive number the overflow flag is set to '1.' The bcd flags indicate that a carry has occurred between nibbles (sets of 4 bits).

Setup panel

The number of bits in the problem can be selected from 1 to 16 with the pull-down menu box.

The mirror enable button allows a mirror of the problem numbers to be displayed in both decimal and hexadecimal form.

Illustration of calculator operation

Fig. 2 shows the calculator after an addition problem has been randomly selected and an answer is partially entered with the mouse.

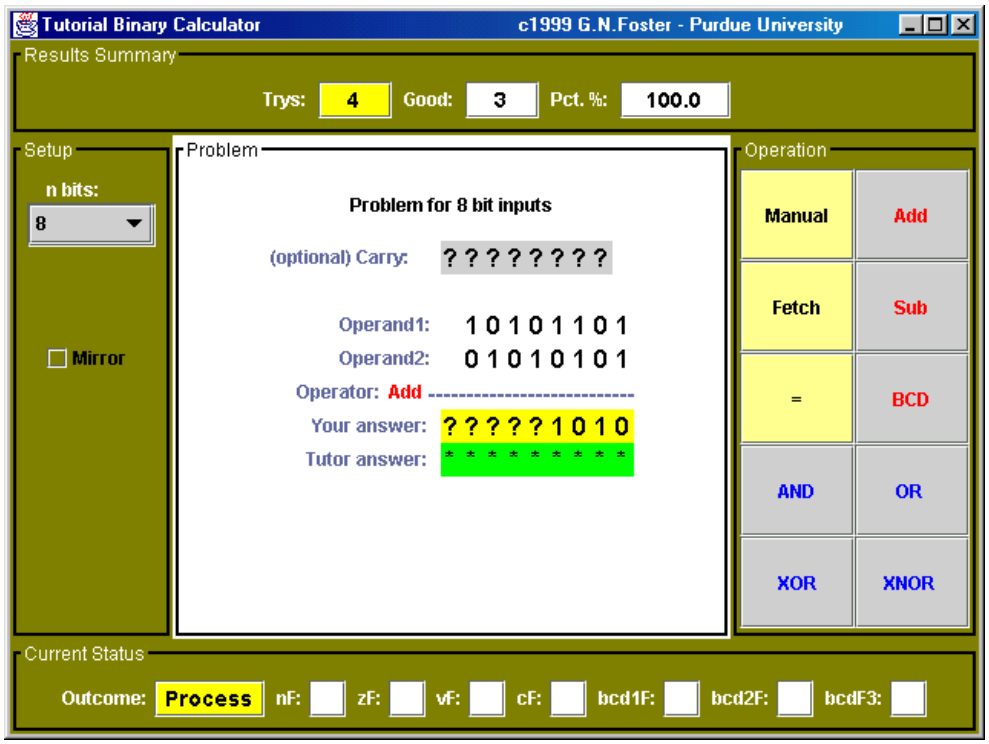

Figure 2: Random fetch of problem values and partial answer.

In the figure above, the Results panel shows that the student answered 3 correct out of 3 attempts and is working on the fourth attempt. The 'Trys' and 'Outcome' displays have yellow backgrounds indicating that the calculator is in the middle of a problem process.

In Fig. 3, the entry of the answer is completed and the equal sign is pressed.

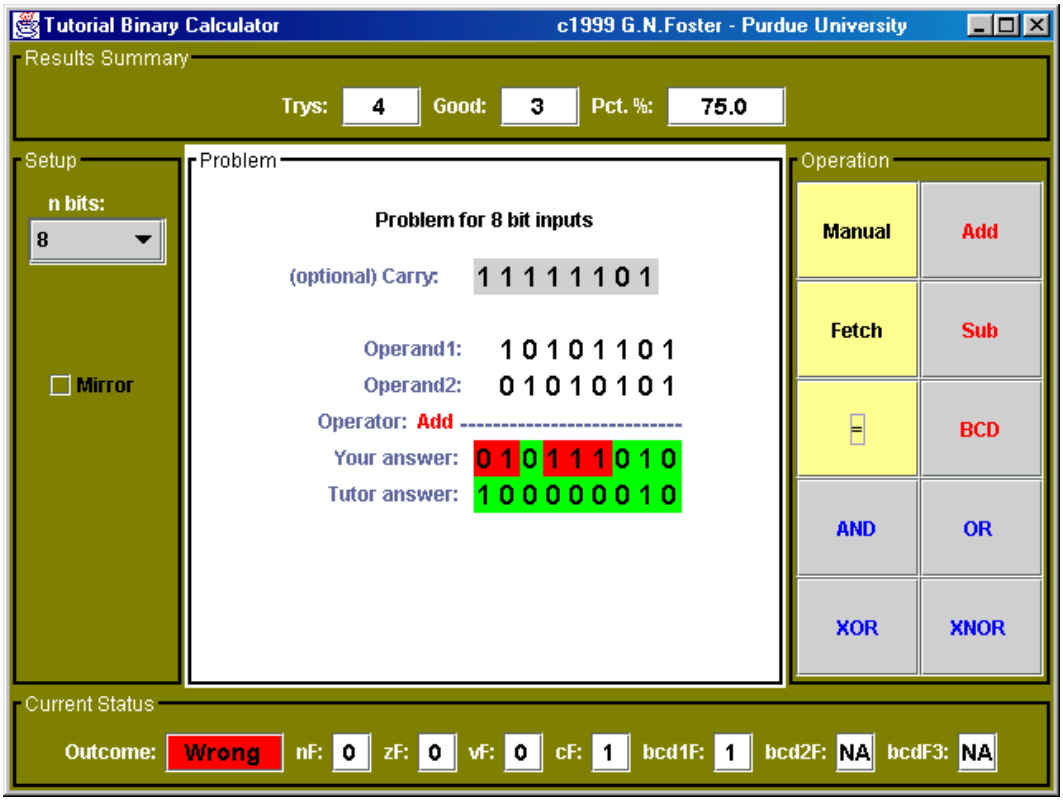

Figure 3: Results of calculator operation

As a result of completing the problem, the calculator displays the correct answer, updates the displays and highlights the correct and incorrect bits of the student's answer in green and red. The display of wrong answers is striking in color. No attempt has been made to include an option for people who are red-green color blind, but that will certainly be addressed in the future.

Figure 4 shows a different configuration where the AND operation is selected with 16 bits.

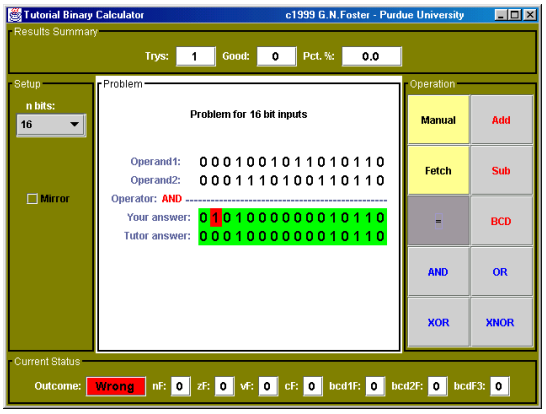

Figure 4: Logical AND of 16-bit numbers.

Software development

The calculator presented was developed with the Java Development Kit (jdk1.2.1) downloaded from Sun Microsystems (java.sun.com or sun.javasoft.com). The author has experience writing code using the object-oriented features of C++ and some experience with JavaScript. The experience with C++ was the more valuable of the two although a number of years has elapsed. Several Java books were read in prior years, but the most helpful ones, the ones cited in the bibliography, were more current and more advanced. At the initiation of this project, May 1999, Java was in version 1.2, which became known as Java 2. In this version, the library of components was expanded into what is called Swing components. Classes in this group have new names. The class Button becomes JButton and so forth, although old classes and names are maintained. Wishing to keep current, the author selected to learn Java 2. Another decision was to learn the raw language and not use a Java editor. The author does not regret these decisions. However, the learning curve is steep especially if one does not know the basics of object-oriented programming.

In some ways, Java is friendlier than C++. Pointers are not allowed so that systems cannot be inadvertently crashed. And Java is more of a straight object-oriented language, without the mixed procedural and OOP slant of C++. After programming in Java, the author has a better "look-and-feel" of object-orientation than before.

The method of implementation was to both learn the language and develop sections of code concurrently. Top-down thought was given to the end product and the books were read. Bottom-up methods consisted of establishing small programming goals and implementing them in code. This approach held together well because object orientation was being used. The CD demonstration programs in the reference books were helpful in providing frameworks. First the bit class was defined. This class had methods that would allow an object bit's background color to be changed in different situations in the dynamic running of the program. The value of a bit could be changed in toggle fashion by clicking a mouse or, in some cases, changed only internally by the program. The next higher class, called 'bitnum', consisted of groups of bits (bit objects) that made up a binary number. The bits were organized in arrays with place values. Methods were written to implement arithmetic or logical operations. Java has its own way of handling events such as button presses and mouse clicks. The flexibility of event handling allows an object to become a listener to the event. Thus a button press can lead to a variety of responses by the program. Many interactive options arise. Beyond these concerns, Java allows a number of Layouts for the display of information (objects) on the screen. The objects to be displayed can be organized into panels and sub-panels. The options in this area can be confusing and time and practice are required to learn this aspect of Java. The development period of learning why something would or would not display with the proper spacing was particularly frustrating and has not been completely mastered. Not withstanding the problems, once the calculator's basic functioning as an adder was completed, the addition of other operations was very easy.

Utilization

At this point, the calculator is in the form of a Java application. It is 50 kbytes long. It has been installed on the G: drive of our LAN server. The complete installation required downloading the JRE (Java Runtime Environment) package from Sun. This software is approximately 5 Mbytes. A batch file invokes the java.exe file to run the calculator. Our advanced electrical laboratory has old 66 MHz 486 PCs with limited RAM. Invoking the software on these systems was not accomplishing in over 5 minutes. The calculator software was called up on the 233 MHz, 64 MB Pentium PCs in the computer laboratory and the calculator display appeared in 15 to 20 seconds.

The first group of students to use this software was a first semester Digital Fundamentals class. They were given a 5-minute introduction to the calculator and then were allowed to use the program in a semi-guided laboratory session. The students were attentive and asked good questions. A number of problems were found. One problem related to how a calculation was faulty for numbers of one bit. Another problem occurred for certain values of overflow in the subtraction operation. However, the responses were favorable. Some of the students wanted to take the software home. One student commented that it would be valuable as a review aid for future courses. On a test, the users of the program appeared to do well on test questions related to the calculator-type of problems.

Conclusion

This work is encouraging. The package is not finished nor are all the links to text accomplished. The advanced features are not installed. But the basic program is working. The display looks polished. The easy of data entry and the display of results provides good feedback to the students. The students' initial reactions were favorable and, by estimating the results of a small sample, learning has been accomplished.

While Java has a fairly steep learning curve, for some people this language will prove useful for creating very flexible and very interactive computer-aided instruction provided that Java maintains a presence in the turbulent application-development, application-delivery wars.

Bibliography

1. Horstmann C.S. & Cornell G. (1999). *Core Java 1.2: Volume 1- Fundamentals*. Palo Alto, CA:Sun Microsystems Press. Pp.742.
2. Horton, I. (1999). *Beginning Java 2*. Birmingham, UK: Wrox Press. Pp. 1110.
3. Van der Linden, P. (1999). *Just Java 2*. Palo Alto, CA: Sun Microsystems Press. Pp. 775.

GERARD N. FOSTER

Gerard (Jerry) Foster is an associate professor of electrical engineering technology at Purdue University, School of Technology at Kokomo, Indiana. He supervises and teaches the digital, microcontroller and digital signal processing sequence of courses. His other interests are in the areas of design, laboratory projects, multimedia, C++ and Java. Professor Foster is current chairman of the Information Systems Division.