8085 Microprocessor Simulation Tool "8085 SimuKit"

ALI CHEHAB, SAMER HANNA, KARIM Y. KABALAN, ALI EL-HAJJ

Electrical and Computer Engineering Department, American University of Beirut, Beirut, Lebanon

Received 11 March 2003; accepted 30 May 2004

ABSTRACT: This paper presents an interactive and user-friendly computer package, "8085SimuKit," which can be used to simulate the operation of an 8085 microprocessor. The package is a practical tool in teaching microprocessor or related courses. The simulator enables the user to verify his/her program in assembly language or directly in machine language. Two built-in editors, one for assembly language instructions and the other for machine language instructions allow the user to type in his code in a similar manner to the integrated development environment offered by other programming languages such as BASIC or C. The simulator will automatically parse the instructions, and extract the commands, operands, and addresses from them. It is also capable of converting an assembly language program to a machine language program, and it gives a list of each assembly command line versus machine code line. The details of registers, ports, interrupts, and flags are all clearly displayed for the user. A copy of the software is available at http://webfea.fea.aub.edu.lb/fea/software/8085simukit.zip. Comput Appl Eng Educ 12: 249–256, 2004; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20022

Keywords: 8085 microprocessor; software tool; simulation; engineering education; assembly language

INTRODUCTION

The enhancement in engineering education can be achieved through software development by utilizing animated graphics of dynamics phenomena and producing pictorial representations of highly abstract mathematical subjects. There are many of such examples in the field of logic systems such as: CAD techniques for circuits with piezoelectric devices [1], CAD Tool for minimizing logic functions [2], and others [3–7].

In most educational and industrial fields that are related to the 8085 microprocessor, engineers or students might face some problems when using 8085 microprocessor hardware simulation boards. For example, if you want to test an assembly program for the 8085up of, say 400 instructions, the first thing you would do is to convert your program to machine language by referring to a conversion table of assembly versus machine. Then, you have to enter your machine code program to the simulation board carefully, and you need to double-check your entry

Correspondence to K. Y. Kabalan (kabalan@aub.edu.lb).

again to avoid mistakes. Finally, after you finish the program testing, you will not be able to save your program and you have to enter it again every time you need to repeat the test. Another problem is that students may not have access to simulation boards when trying to do their homework or when working on their projects. Hence, there is a need for a flexible solution whereby engineers or students can reliably test their 8085up programs and have the opportunity to save their programs, to test them at the assembly language level, and to be able to do it at home, and not necessarily in a laboratory. Therefore, a convenient solution to overcome the problems behind using 8085up simulation boards is to introduce simulation software for 8085up running under the environment of Microsoft Windows.

The "8085SimuKit" is an 8085up simulation tool that works under windows (9x, Me, 2000, or XP). This software enables you to verify your program in assembly language or directly in machine language. Users may benefit from the built-in editors that allow them to edit their code just like they do in the integrated development environment of other programming languages such as BASIC or C. The simulator will automatically parse every instruction and extract the commands, operands, and addresses from it. It is also capable of converting the assembly language code to machine language code and it gives a list of each assembly command line versus machine code line. The details of registers, ports, interrupts, and flags are all clearly displayed for the user. Saving the program in assembly or machine code is not the only possibility of this kit, but it also the user to save the input port setting. Moreover, the user may choose to execute his/her program in different ways such as complete execution of the code at once, or step-by-step execution. This software covers all of the 8085up known instructions.

SOFTWARE DESCRIPTION

Figure 1 shows the main window of the "8085SimuKit." As it can be seen, the left-hand side is divided into two parts: the upper part is a machine language editor while the lower part is an assembly language editor. The right-hand side of this window shows the complete registers list, flag register details, input/output port setting window, software controlling buttons, interrupt inputs, serial output/input data

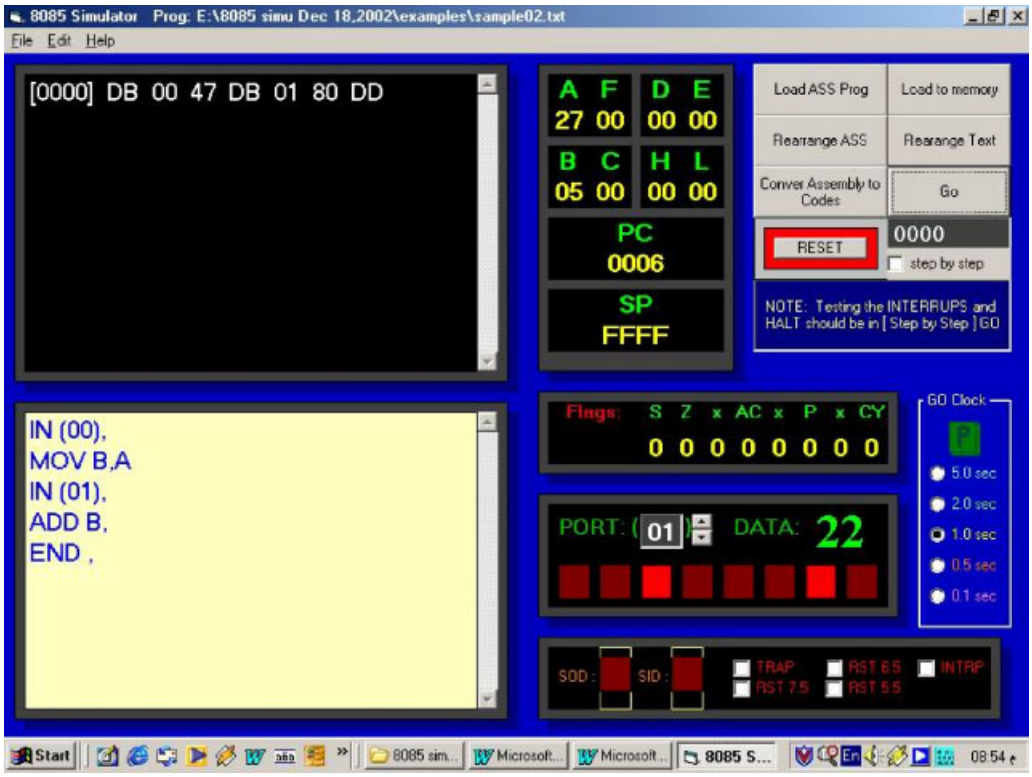

Figure 1 "8085SimuKit" main menu. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

SOD/SID, and the controlled execution options. Next, we describe the functions and usefulness of these various parts.

THE MACHINE CODE WINDOW

The machine code window, which is shown below in Figure 2, is the part of the software where users can enter their programs in machine language. To do so, the following rules should be taken into consideration.

(a) Each code should have a size of one byte (3E 23 06 0E 00 00 16 77 DD).
(b) Consecutive codes are to be separated with spaces (3E 23 06 and not 3E2306).
(c) Addresses are to be written between square brackets ([10FE] 3E 44 DD).
(d) If your program does not contain any address, then the default starting address will be considered as [0000] and consecutive codes will occupy consecutive memory locations.

For more illustration, several scenarios are considered. If the user enters into the machine code window "3E FF 06 CA 00 DD," then the codes 3E, FF, 06, CA, 00, and DD will be stored in memory locations [0000], [0001], [0002], [0003], [0004], and [0005], respectively. If the user enters "3E FF C3 0A 00 [000A] 3E 12 DD [1003] 3E 32 DC DD," then the codes 3E, FF, C3, 0A, 00, 3E, 12, DD, 3E, 32, DC, and DD will be stored in memory locations [0000], [0001], [0002], [0003], [0004], [000A], [000B], [000C], [1003], [1004], [1005], and [1006], respectively.

Figure 2 The machine code window. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

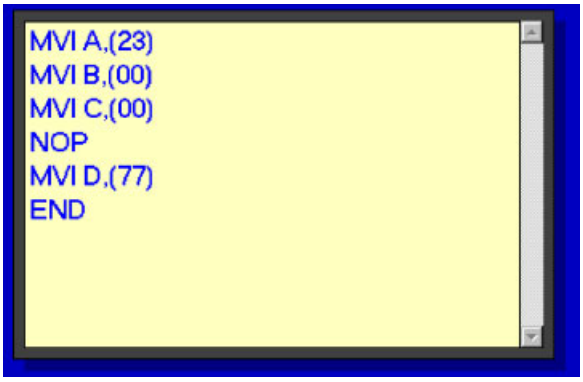

Figure 3 The assembly window. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

THE ASSEMBLY WINDOW

In the assembly window, the user can enter the assembly language program line by line in the format shown in Figure 3.

When using the assembly window, the following rules for program writing should be taken into consideration.

(a) Addresses are to be written between square brackets.
(b) Operands are separated with commas.
(c) There must be a space between the command part and operands.
(d) Data bytes are written between parentheses.
(e) Although there is no "END" command in the 8085up instructions, the user must indicate the end of the program with an "END" command.

The assembly language program in Figure 4 illustrates the above-mentioned points.

THE INPUT/OUTPUT PORT WINDOW

The input/output port window is shown in Figure 5. In this window, the user can set the input port data or view data at an output port. To set data on a specific

```
[000A]
MOV A,B
ADD C
MVI D,(23)
JMP [132F]
END
```

Figure 4 Assembly language program example.

Figure 5 The input/output port window. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

port, first select the port address where the data is to be set by writing its number in the Port Address Field directly or by using the Increment/Decrement buttons to increase or decrease the port address. Then, click on the Data Bit Buttons to set the value applied on that address. The data you set will be also shown in Hexadecimal. This window can also be used to show the data at an output port.

Figure 6 explains the function of each of the bottoms of this window.

THE REGISTERS WINDOW

This window, which is shown in Figure 7, gives the data values of the registers in hexadecimal. They are arranged in pairs, as they actually exist in the 8085 microprocessor. Since the flag register bits refer to different details, "8085 SimuKit" gives a detailed window for the flag register as shown in Figure 8.

In Figure 8, S is the sign bit, Z is the zero, AC is the auxiliary carry, P is the parity, and CY is the carry.

DATA AND INTERRUPT WINDOW

In this window, and as it can be seen in Figure 9, the user can either set the serial input data "SID" by clicking it, or view the value of the serial output data "SOD." All 8085up interrupts are available in this software. The priority of interrupts is important in a way that if the user clicks "RST7.5," then clicking "RST6.5" or "RST5.5" will be ignored in accordance with the interrupt priority. Note that the execution of the program should be done in a step-by-step mode in order to test the interrupts.

ACCESS CONTROL BUTTONS

The access control buttons are shown in Figure 10. This window is used to load the program from the editor window into the memory and rearrange the command lines. The functions of each of the buttons keys in Figure 10 are as follows:

(a) *Load ASS Prog*: This key allows the user to load the entered assembly program into the assembly memory.
(b) *Rearrange ASS*: This key will rearrange the entered assembly program. This key is activated after the program is loaded into the assembly memory.
(c) *Convert Assembly to Codes*: This key will convert the assembly program to a machine language program.
(d) *RESET*: This key resets the values of registers and flags.

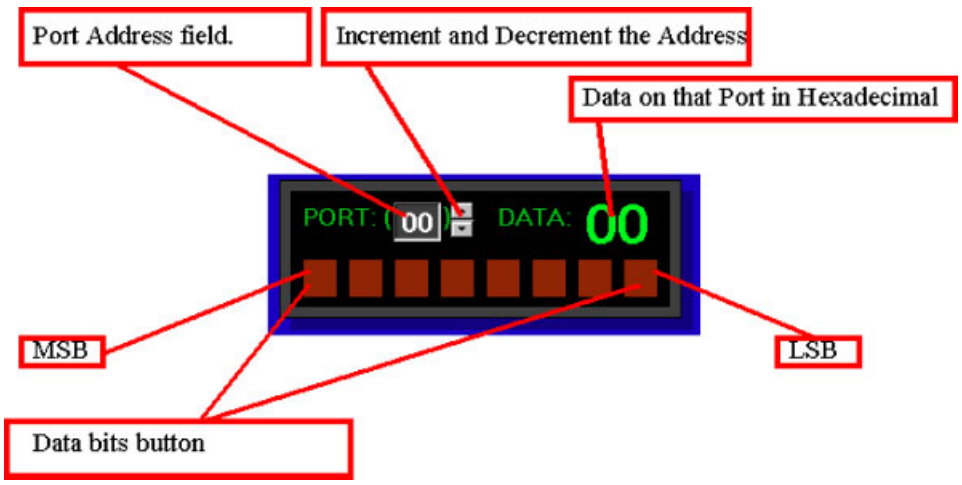

Figure 6 Explains the function of each of the bottoms of this window. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

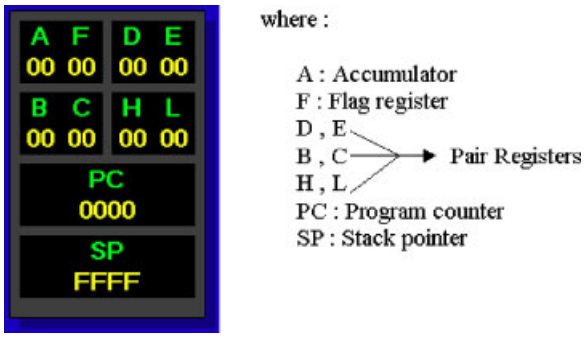

Figure 7 The registers window. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

(e) *Load to Memory*: This key loads the machine language program to the main memory. It acts like the external RAM in the 8085up board where machine codes can be stored.
(f) *Rearrange Text*: This key is used to rearrange the machine language program. This key is activated after the machine program is loaded into the memory.
(g) *GO*: This key is used to execute the machine language program.
(h) *[0000]*: This indicates the memory address of the command being currently executed. The user can change this field manually to any address value.
(i) *Step by Step*: This option allows a step by step execution of the program. When this option is selected, individual instructions are executed one at a time as we click the "GO" button.

THE GO CLOCK

In the "step by step" mode, instead of clicking on the "GO" button in order to advance the execution of the program, the user can alternatively automate this process by specifying the desired time duration that separates the execution of the consecutive instructions in the "GO Clock" window, and then by pressing the "P" play button once to activate, and once again to

Figure 8 The flags window. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Figure 9 The serial data and interrupt window. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

stop. There are five different time durations available in this window as shown in Figure 11.

THE MENU BAR

The menu bar consists of the "File," "Edit," and "Help" menus. The "File" menu contains the following sub-menus: "Open," "Save," "Save as," "Exit," and "save/open" port settings. The "Edit" menu contains one option that gives a list of the assembly command lines corresponding to the machine code. The "Help" menu contain the quick user guide written in word, and an option about this program Figure 12.

HOW TO PROGRAM

In this section, general instructions are given to guide the user on how to verify a simple program in assembly language and how to carry out its execution.

(a) Write the program in the Assembly Language Editor Window.
(b) Load it to the assembly memory by clicking "Load ASS Prog."

Figure 10 The access control buttons. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

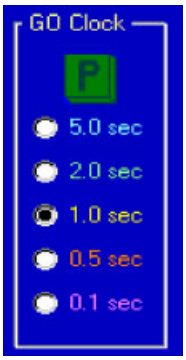

Figure 11 The "GO" clock. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

(c) Convert it to machine language by clicking the "Convert ASS to Codes." The corresponding code will then be shown in the Machine Code Editor Window.
(d) Load it to the main memory by clicking on the "Load to memory" button.
(e) To run the program, just click on the "GO" button. This will result in the execution of the complete program at once. For a step-by-step execution, just click on "Step by Step" check box so that this will change the "GO" button to "Step by Step GO." Every time you click on it, an individual command line will be executed. Note that the Start Address in the Address box will be changed on each click to reflect the new memory address of the instruction currently being executed. Therefore, if you want to run the program again, change the starting address to its initial value in the Address Box. Another option for executing the program is by using the "GO clock" window whereby you can select a specific timer. Instead of continuously pressing the "GO" button, the execution of the instructions will be advanced according to the time interval you have specified in the "GO CLOCK" window. So, first you select the step-by-step option, then you select the desired time interval and you click the "P" play button. Finally, you click the "P" button once again to stop the execution of the program.

Figure 12 The menu bar. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

In addition to the above-mentioned features, the user can view interactively the changes in the values of registers, interrupts, and flag bits as well as the value of the program counter (PC) as the execution of the individual instructions advances.

The machine code command lines are shown below in Figure 13, along with the corresponding PC values.

For example, in Figure 13, the user can see that the initial address was [0055] which refers to the start of "3E 12" command line. The values of initial registers, interrupts, and flag details are shown also at that moment.

To activate the run-time analyzer, check "Activate Analyzer" in the "Edit" menu before you execute the program by clicking "GO." Then click "show analyzer screen" in the "Edit" menu to view the analyzer screen. The "Activate Analyzer" is unchecked when you click "Reset" or when you load or open a new program.

COMPARISON WITH OTHER AVAILABLE SIMULATORS

Comparing the "8085SimuKit" with other available software, the "8085SimuKit" offers three main features that other 8085 simulations lack (Table 1). The three features are the Assembly and the Machine code editors, the interrupts testing, and the I/O ports support. Comparison is made with "uPsim version 1.12" [3], "up8085 Simulator" [4], "8085 Simulator BubbleSORT 2.85" [5], and "8085 under DOS" [6].

First, the "8085SimuKit" allows the user to enter the assembly language instructions or machine codes in textual rather than tabulated format. This feature is not available in the other software. For example, in Reference 8, there is a pad of assembly commands; so that, in order to write your assembly instructions you have to click the instructions and operands on the

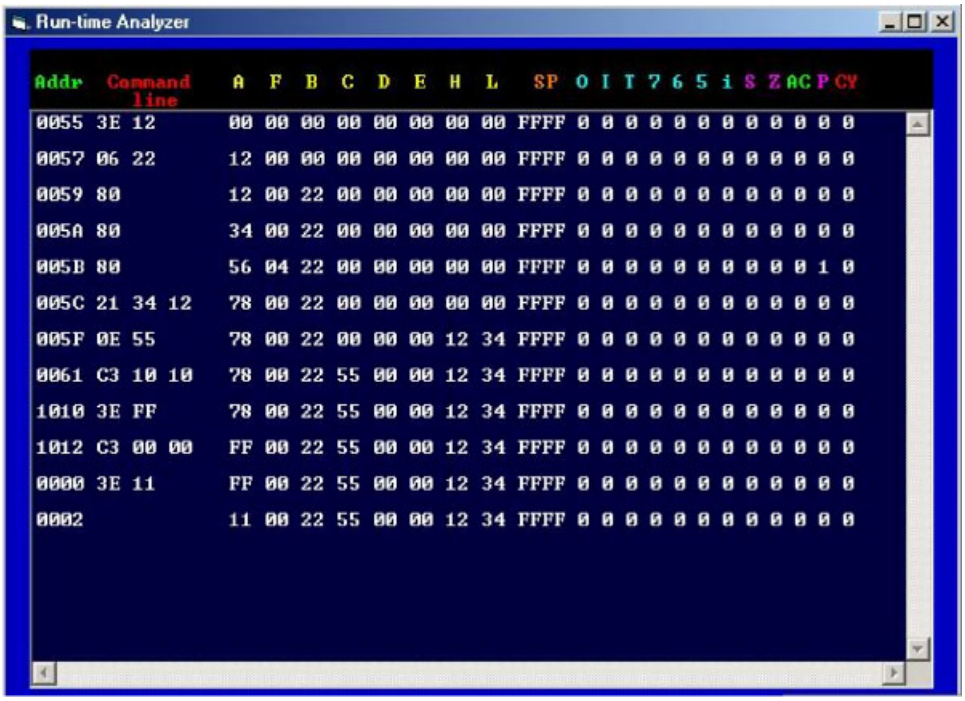

Figure 13 The run-time analyzer form. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Table 1 Comparison of Different 8085 Simulation Tools

Point to compare	"8085SimuKit"	"uPsim Ver 1.12"	"up8085 Simulator" by Pinkesh Creation's	"8085 Simulator BubbleSORT2.85"	"8085" (under DOS) by V. Kumar
Assembly editor	Available	Not available	Does not support Assembly language	Available	Does not support Assembly language
Machine code editor	Available	Not available	Not available	Not available	Not available
Interrupt testing	Available	Not available	Available	Not available	Not available
I/O ports	Available with full range of 256 port addresses	Not available	Available with range of only seven port addresses	Not available	Not available

instruction pad list. The other two simulators do not support 8085-microprocessor assembly language; and they allow the user to program in machine code only. To build your program in References 9 and 11, the user has to click the memory location from the list of memory locations and then enter the code. In Reference 10, the user can load only an assembly language program and not a machine language program.

Second, the "8085SimuKit" allow the user to verify interrupt requests during program execution. In all of the above-mentioned software, except Reference 10, this feature is not supported.

Third, the "8085SimuKit" offer the user with the complete range of 256 input/output ports unlike Reference 8 where they are limited to only seven ports. Moreover, in "8085SimuKit, the user can view the port setting in both bit format as well as hexadecimal format.

CONCLUSION

The CAD package described in this paper is an interactive, user-friendly and provides a practical tool for teaching microprocessor or related courses. Its capabilities includes simulating a whole set of instructions, allowing interrupt requests, and input/output port testing of the 8085 microprocessor. The user can easily install, understand, and make use of the various features that are supported by this software package.

REFERENCES

[1] R. W. Rhea, CAD for circuits with piezoelectric devices, Proceedings of the Annual IEEE International Frequency Control Symposium, 2000, pp 250−254.
[2] K. Y. Kabalan, A. El-Hajj, S. Fakhreddine, and W. S. Smari, Computer tool for minimizing logic functions, Comput Appl Eng Educ 3 (1995), 55−64.
[3] K. Mladen, Advanced education and training using new digital simulator designs Proceedings of the American Power Conference, Vol. 59-1, 1997, pp 481−486.
[4] J. David Jeff, Hierarchical digital systems modeling utilizing hardware description languages for computer engineer education, Comp Electr Eng 21 (1995), 311−320.
[5] H. Lawrence and R. Charles, ''Teaching digital system design with a multilevel digital systems simulator'' Proceedings—Frontiers in Education Conference, 1987, pp 30−36.
[6] I. Longair, Digital filter—an interactive computer program for the design and simulation of a finite impulse response (F.I.R.) digital filter, Int J Elect Eng Educ 23 (1986), 339−348.
[7] K. George, Digital filter simulation in the classroom, Modeling and Simulation, Proceedings of the Annual Pittsburgh Conference, Vol. 21, Computers, Computer Architecture, and Microprocessors in Education, 1990, pp 1269−1273.
[8] http://www.geocities.com/ransandanks/
[9] http://www.angelfire.com/in3/myweb/mic8085.html
[10] http://www.ocf.berkeley.edu/∼amanb/oldhtml/8085.html
[11] http://ping-systems.com/

BIOGRAPHIES

Ali Chehab received his bachelor's degree in electrical engineering from the American University of Beirut (AUB) in 1987, the master's degree in electrical engineering from Syracuse University, and the PhD degree in electrical and computer engineering from the University of North Carolina at Charlotte, in 2002. From 1989 to 1998, he was a lecturer in the Department of Electrical and Computer Engineering at AUB. He rejoined the department at AUB as an assistant professor in 2002. His research interests are VLSI design and test and the development of educational software tools.

Samer Hanna was born in Baghdad, Iraq, in 1976. He received his bachelor's degree in electronics and communication engineering from Baghdad University in 1998 and a master's degree in engineering in computer and communication engineering from the American University of Beirut in 2004. From 1999 until 2004, his work experience involved power generators, medical equipment, and computer networks development/training and technical support. His research interests include mobile agents and simulation programs.

Karim Y. Kabalan was born in Jbeil, Lebanon. He received the BS degree in physics from the Lebanese University in 1979 and the MS and PhD degrees in electrical and computer engineering from Syracuse University in 1983 and 1985, respectively. During the 1986 fall semester, he was a visiting assistant professor of electrical and computer engineering at Syracuse University. Currently, he is a professor of electrical and computer engineering with the Department of Electrical and Computer Engineering, Faculty of Engineering and Architecture, American University of Beirut. His research interests are numerical solution of electromagnetic field problems and software development.

Ali El-Hajj was born in Aramta, Lebanon. He received the license degree in physics from the Lebanese University, Lebanon, in 1979, the degree of *ingenieur* from L'Ecole Superieure d'Electricite, France, in 1981, and the *docteur ingenieur* degree from the University of Rennes I, France, in 1983. From 1983 to 1987, he was with the Department of Electrical Engineering at the Lebanese University. In 1987 he joined the American University of Beirut, where he is currently professor of electrical and computer engineering. His research interests are numerical solution of electromagnetic field problems and engineering education.