A Functional Based Simulator for the 8051 Microcontroller

Ahmet Turan Ozcerit
Sakarya University, Technology Faculty,Sakarya, Turkey
aozcerit@sakarya.edu.tr

Necat Guney
Izzet Baysal Vocational High School, Bolu, Turkey
necatguney@hotmail.com

Abstract: In this paper, a functional based simulator has been designed to facilitate the education of the 8051 microcontroller, which is used widely in today's engineering and educational purposes. In the realization of the simulator, a PC-based program is developed in a way to make easy to follow the contents of the registers and flags of the running an 8051-coded assembly program. Another important contribution fulfilled is a toolbox, which can easily be connected to the ports of the microcontroller using as switch, button, display, etc. The units in the toolbox can also be connected after compilation period. By the help of this facility, the simulator can be used as a virtual microcontroller development board. Such properties simplify to understand the internal architecture of the 8051 microcontrollers for students and designers.

Introduction

Microcontrollers, apart from microprocessors, can handle many industrial tasks in effort free manner and they are mostly proper answer for tight budgeted projects. They can be frequently called as embedded computer since they can be used as a single chip for industrial solutions. Reprogramming property of these chips attracts many designers to overcome various practical applications along with electrical control purposes.

Electronics hobbyists and engineering students are also primary target groups for embedded microcontroller chips. Many electrical or computer engineering departments from all over the world include microcontrollers in their course curriculum along with assembly programming. Since these courses mostly cover practical side of the microcontrollers, an experimental based laboratory is required. However, experimental laboratories are not appropriate solution when financial or spatial shortcomings are inevitable (Smith M. R., Cheng M, 1996) . Using software-based simulators can offer affordable alternatives to hrdware based experimental kits. Nevertheless, there are some limitations for almost every simulator, for example, simulators cannot always run in real-time mode, neither debugs timing problems of the real system (TOPALOGLU N., 2002).

Related Works

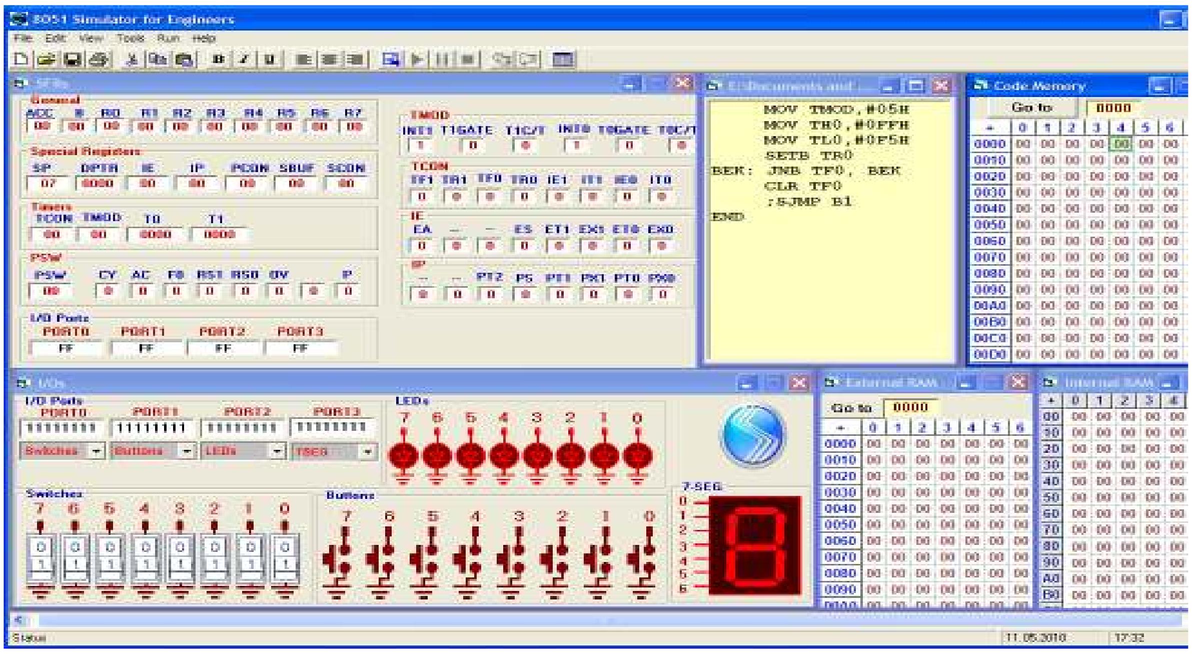
In the past, many microprocessor simulator tools were designed and used by commercial purposes. However, there are some examples were designed by educational and academic purposes. For example, C. W. Caldwell et al. (Caldwell C., et al., 1995) proposed a graphical microprocessor simulator to be used in engineering classes. By using their simulator, the students could also reach to see internal registers and computational details of the microprocessors. Another simulator was designed for 68HC11 microcontroller at South Carolina University in microcontroller programming courses for the students attending mechatronics engineering department. Students could monitor each minute detail of the microcontroller such as RAM, ROM contents, ports, serial interface, timers, etc. This simulator also enabled students to debug the designed system. At the end of the course, many students focused on industrial applications that could be completed by microcontrollers (Giurgiutiu V., et al., 2005).

In another paper, interpreter-based and compiler-based simulators were united in one simulator in order to utilize the best features of both techniques (Reshadi M., et al., 2003) Interpreter-based simulators are preferred for flexibility and detailed operations while compiler-based simulators are used for fast simulations. The simulator designed blended both simulators to increase the flexibility and performance of the operations to balance each parameter. The simulator we designed has aimed to teach students the use of instruction set and assembly programming of 8051 microcontrollers.

The Component Details of the Simulator

The 8051 simulator we designed has eight windows as seen in Figure-1:

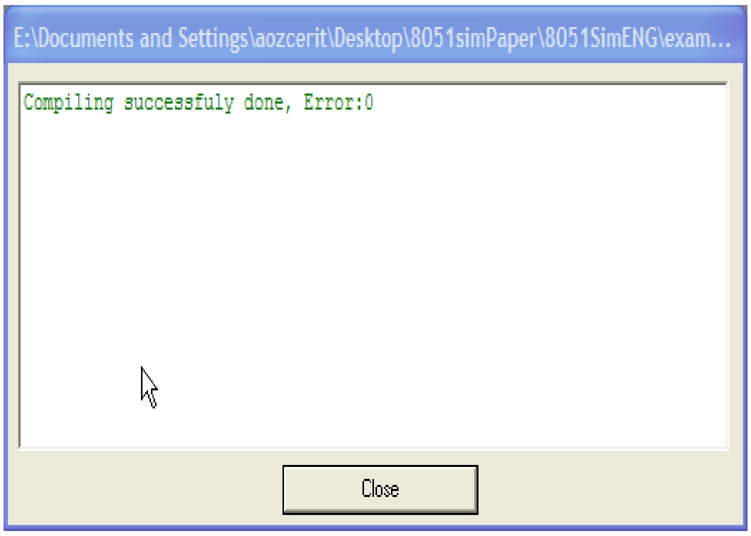
- Code editing window
- Code compiling and error message window
- Program run control window
- Program code window
- Internal RAM window
- External RAM window
- Special Function Registers (SFRs) window
- Input/Output ports window



```
        MOV  TMOD,#05H
        MOV  TH0,#0FFH
        MOV  TL0,#0F5H
        SETB TR0
BEK:    JNB  TF0, BEK
        CLR  TF0
        ;SJMP B1
END
```

Figure-1: Main components of the 8051 simulator

Code editing window can be used either for new codes or for the codes loaded from the disk. In this window, the program can be entered in 8051 assembly language. Once the program is assembled, a window is appeared to display assembler result as indicated in Figure 2.



E:\Documents and Settings\aozcerit\Desktop\8051simPaper\8051SimENG\exam...
Close

```
Compiling successfuly done, Error:0
```

Figure 2. Compile window showing assembler results

At this stage of the simulator, we have used standard ASM51.exe to assembly target assembly coded files. Entire hex code, list code program development cycle is illustrated in Figure 3. Hex files can either be used in many simulator programs, emulators or can be downloaded into a target 8051 derivative chip by programmer software or a programmer device.

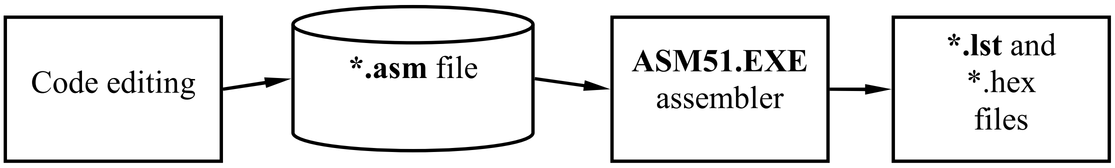
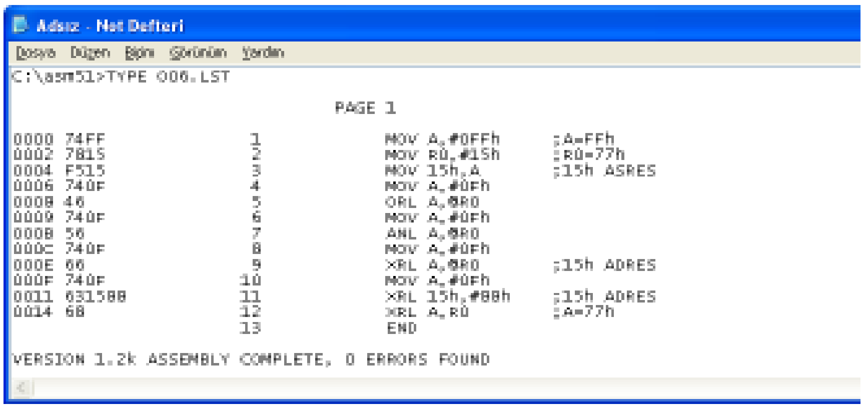


Figure 3. Development cycle of hex and list files

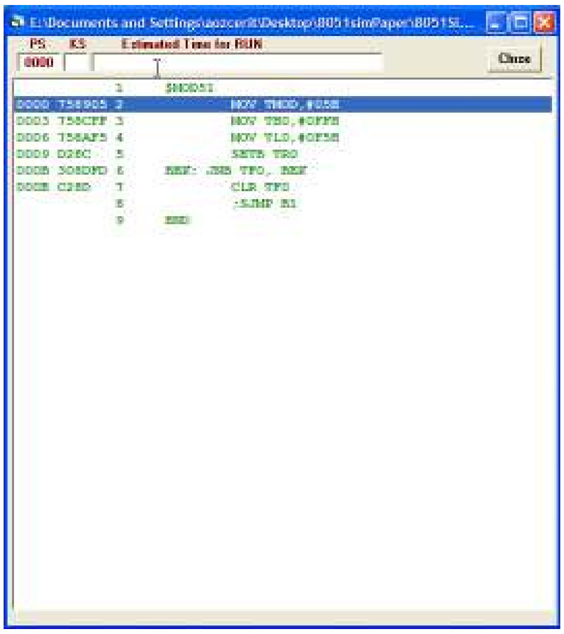
In the list file, program assembly codes, corresponding hex codes, and code memory addresses are created as seen in Figure 4. While the first column represents the memory address of corresponding assembly codes, the second is for opcodes of that assembly codes. The simulator we designed uses first, second and third columns to accomplish desired code simulation.



```
C:\asm51>TYPE 006.LST

                                        PAGE 1

0000 74FF                  1            MOV A,#0FFh       ;A=FFh
0002 7815                  2            MOV R0,#15h       ;R0=77h
0004 F515                  3            MOV 15h,A         ;15h ASRES
0006 740F                  4            MOV A,#0Fh
0008 46                    5            ORL A,@R0
0009 740F                  6            MOV A,#0Fh
000B 56                    7            ANL A,@R0
000C 740F                  8            MOV A,#0Fh
000E 66                    9            XRL A,@R0         ;15h ADRES
000F 740F                 10            MOV A,#0Fh
0011 631588               11            XRL 15h,#88h      ;15h ADRES
0014 68                   12            XRL A,R0          ;A=77h
                          13            END

VERSION 1.2k ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

Figure 4. List file contents

If the assembly file includes errors, a set of error messages are issued. Having a successful compilation completed, the assembly program can be run in step-by-step (F8) or in full speed (Ctrl + F12) fashion on run window as shown in Figure 5. On this window, a program counter, a cycle counter for current instruction, and total simulation timer are deployed to monitor simulation events.

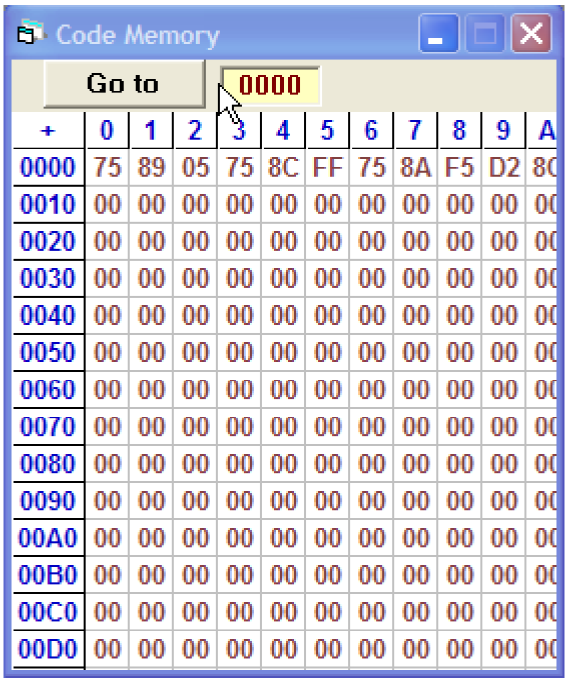

E:\Documents and Settings\aozcerit\Desktop\8051simPaper\8051Si...
PS
KS
Estimated Time for RUN
0000
Close

```
                1      $MOD51
0000 758905 2                MOV TMOD,#05H
0003 758CFF 3                MOV TH0,#0FFH
0006 758AF5 4                MOV TL0,#0F5H
0009 D28C   5                SETB TR0
000B 308DFD 6      BEK: JNB TF0, BEK
000E C28D   7                CLR TF0
                8                SJMP B1
                9      END
```

Figure 5. Program run control window

In the simulation stage, the internal and external memory blocks can be monitored. The internal RAM window incorporates direct and indirect data memory and it can be modified by the user in bit or byte manner during the simulation. However, the simulation needs to be paused before the modification. Any change in internal RAM cell can be noticed immediately since the related RAM cell is colored in real-time. This facility enables the user to control the simulation efficiently.

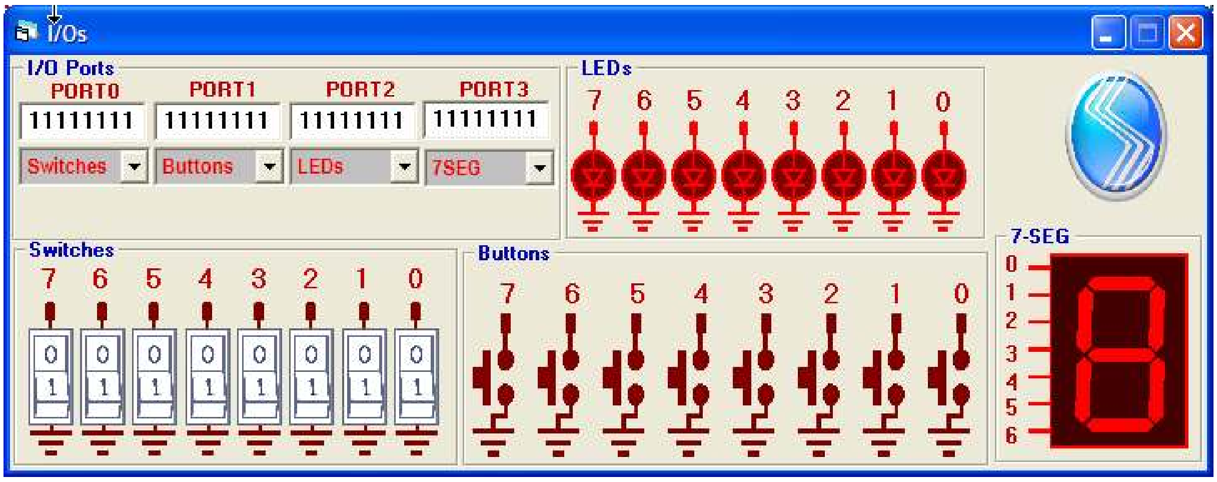
The external RAM and SFR memory windows have identical functions what internal RAM window has. However, program code memory represents the hex code of the assembly programs as seen in Figure 6. The program codes normally cannot be modified by the users, but if some users who are very confident on what they do can modify the contents of the code memory in the paused mode.



+	0	1	2	3	4	5	6	7	8	9	A
0000	75	89	05	75	8C	FF	75	8A	F5	D2	8
0010	00	00	00	00	00	00	00	00	00	00	0
0020	00	00	00	00	00	00	00	00	00	00	0
0030	00	00	00	00	00	00	00	00	00	00	0
0040	00	00	00	00	00	00	00	00	00	00	0
0050	00	00	00	00	00	00	00	00	00	00	0
0060	00	00	00	00	00	00	00	00	00	00	0
0070	00	00	00	00	00	00	00	00	00	00	0
0080	00	00	00	00	00	00	00	00	00	00	0
0090	00	00	00	00	00	00	00	00	00	00	0
00A0	00	00	00	00	00	00	00	00	00	00	0
00B0	00	00	00	00	00	00	00	00	00	00	0
00C0	00	00	00	00	00	00	00	00	00	00	0
00D0	00	00	00	00	00	00	00	00	00	00	0

Figure 6. Code memory window

I/O ports, as illustrated in Figure 7, can be connected to a couple of hardware-based devices that are used frequently for embedded systems. Each port can be connected to LEDs, switches, buttons, or seven-segment display. At reset time, each port is connected predetermined devices, but if desired, all ports can be left alone. Each port's connection can be reorganized during the simulation. There are some constraints though, for example, at any time, two ports cannot be connected to the same device.

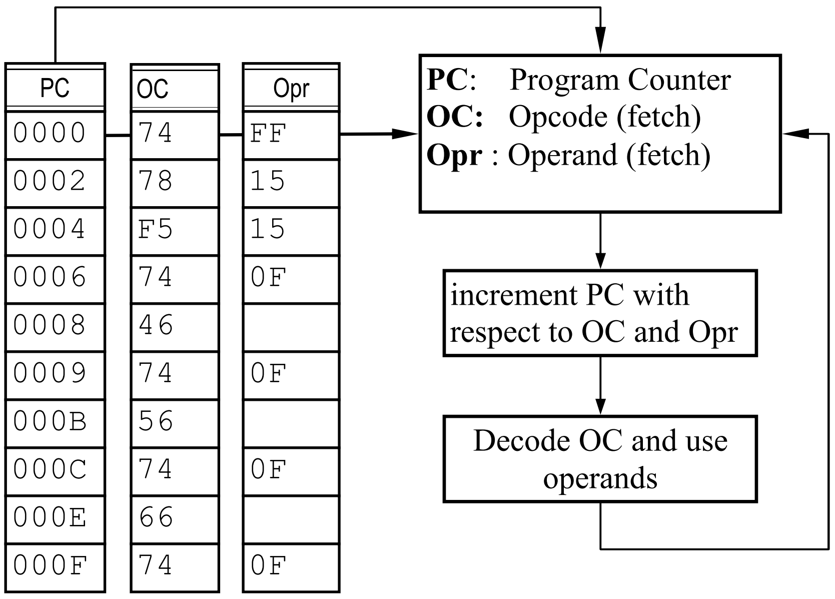


PORT0	PORT1	PORT2	PORT3
11111111	11111111	11111111	11111111
Switches	Buttons	LEDs	7SEG

Figure 7. The I/O ports window

The Fundamentals of Simulation Operations

Once the errors are corrected, program counter value and its related opcode and operands are inserted into arrays. The functional setup can be seen in Figure 8.


PC: Program Counter
OC: Opcode (fetch)
Opr : Operand (fetch)
increment PC with respect to OC and Opr
Decode OC and use operands

PC	OC	Opr
0000	74	FF
0002	78	15
0004	F5	15
0006	74	0F
0008	46	
0009	74	0F
000B	56	
000C	74	0F
000E	66	
000F	74	0F

Figure 8. Block diagram of the simulator

In our design, we have used the program counter as array indexer. The PC is increased automatically according to the type of the instruction. The simulation continues until an *END* instruction is encountered. In order to separate opcodes from *END* instruction, we have used A5 hex number, which is not compatible with any opcode in 8051 assembly language (MacKenzie, I.S.1995).

Conclusions

Microprocessor and microcontroller courses often require experimental laboratory applications. However, the students cannot always use hardware-based systems efficiently. Additionally, such systems can bring major drawbacks such as financial or physical space difficulties. Students can have a great opportunity to learn the course contents if software based solutions are provided. As an alternative solution, we have designed and implemented a free and functional-based 8051-microcontroller simulator to be used in microprocessor classes. The students not only create and edit their assembly codes in the provided environment, also can utilize hardware-based devices. So, the risk of malfunctioning real hardware devices can be avoided. However, the simulator we designed cannot run in real-time mode. This property will be improved in the next versions.

Acknowledgements:

The authors wish to acknowledge the financial support of Sakarya University entitled BAPK Project No. 2010-50-01-010

References

Smith M. R., Cheng M. (1996) "Use of Virtual (simulated) hardware devices in microprocessor laboratories and tutorials", Frontiers in Education Conference, FIE'96, 26th Annual Conference − 1996 − Vol. 3. − Pages 1181–1185.

TOPALOGLU, N. (2002) "The Design and Implementation of PC-Based Functional Microprocessor Simulator", PhD. Thesis, Ankara, 2002

Caldwell, C. W., Andrews, D. L., and Scott, S. S. (1995) "A Graphical Microcomputer Simulator for Classroom Use" Proceedings of the Frontiers in Education Conference, 1995. Proceedings, Vol. 2, Pages: 3b3.9-3b3.12, ISBN:0-7803-3022-6

Giurgiutiu, V., Lyons J, Rocheleau D, Liu, W. (2005) "Mechatronics/ microcontroller education for mechanical engineering students at the University of South Carolina", Columbia, Mechatronics, Volume 15, Issue 9, 2005, Pages. 1025–1036.

Reshadi, M.,Mishra, P., Dutt, N., Proceedings of the 40th annual Design Automation Conference, "Instruction Set Compiled Simulation: A Technique For Fast And Flexible Instruction Set Simulation", 2003, Pages: 758–763, Anaheim, California, USA, ISBN:1-58113-688-9

I.Scott MacKenzie, (1995), The 8051 Microcontroller, Prentice Hall