

A Phenomenon Approach towards Simulation Analysis of Microprocessor 8085

Garima

*Department of Computer Engineering
G.I.T.M., Bilaspur, Haryana, India*

Preeti

*Department of Computer Engineering
Banasthali, India*

Arun patel

*Department of Computer Engineering
G.I.T.M., Bilaspur, Haryana, India*

Abstract- A simulator is defined as any device or system that simulates specific conditions or the characteristics of a real process or machine for the purposes of research or operator training. A simulator is the one that simulates, especially an apparatus that generates test conditions approximating actual or operational conditions. Simulator will be of utmost help to students who are willing to learn more about 8085 programming but don't have the hardware to help them. The simulator we are providing is free of cost and can be employed in any number by the various institutions to train the engineers and other students. As these simulators are far much advanced than the kit we have been using until now, for example we have saving options in these simulators, we have stepwise execution, we can run and compile our programs on a single click instead of remembering the sequence of instructions to follow and these simulators are very less error prone as compared to the kit we are using, the Simulator can be put to commercial use, if one deals with 8085 Microprocessor. One more feature that kind of irritates programmers while working with the kit is the need of opcodes. The kit supports only the opcodes and not the mnemonics. The programmer spends a considerable amount of time in mapping the mnemonics with their respective opcodes. This time is totally a waste for the programmer as these opcodes do not have a programming significance. Thus a good simulator allows us to write our program directly in assembly language instead of opcodes, which saves a considerable amount of time which can be utilized in a better way and removes the problems and the inconveniences faced by the programmer while mapping each and every mnemonic with its respective opcode.

Keywords – Simulator, microprocessor 8085, opcodes.

I. INTRODUCTION

Presently we are using M85-01 kit in various institutions to train engineers regarding the programming with assembly language. The kit is single board microprocessor kit. The kit has been very helpful for training purposes as it provides us with various features, but as everything has some loopholes, the kit also has various disadvantages like

- Since we are working with hardware there is always a chance that it may fail.
- The keypad allows us to enter only hexadecimal values. That forces us to manually determine opcode for all the instructions before proceeding. That can be tiresome.
- It is not possible to view the status of all the registers simultaneously.

In order to overcome the above difficulties, this software solution is developed that is very user friendly and operationally similar to the hardware counterpart. A simulator is defined as any device or system that simulates specific conditions or the characteristics of a real process or machine for the purposes of research or operator training. A simulator is the one that simulates, especially an apparatus that generates test conditions approximating

actual or operational conditions. We have developed a software that will overcome all the difficulties faced by the programmer while using the kit. The software is a standalone application developed in JAVA which will very easily solve all our problems, within a very nominal cost. The simulator has following things in the interface:

Mnemonic Code Table: A mnemonic code table is provided where the user can enter the program directly in the assembly language. The opcodes will be entered automatically by the software making the task very easy for the programmer.

Instruction Chart: A chart containing all the instructions is provided in the software interface from where the user can select the instructions he wants to execute.

Operands Chart: A chart contain various operands is also provided from where the user can select the operands. The chart also contains the set of instructions.

Register: All the registers are shown in the register table .All the changes occurring in the registers during program execution can be seen in this table.

Flags: The flag changes can be seen in the flag register table directly in the simulator.

Origin: We can explicitly specify the origin of the program by entering the address in the origin textbox.

Now let's have a look at all the difficulties and see how the software will overcome those difficulties.

The first and foremost difficulty we faced was the portability. It was difficult for us to carry the kit from one place to another whereas the software can be very easily transferred from one place to another using any device.

The second problem we faced was the memory. A large amount of memory was not available and we could not see all the memory locations simultaneously whereas in this software we have a memory map which allows us to see all the memory locations simultaneously at a single glance.

The third problem we faced was the display. A very small 7 segment display is available on the kit whereas here we have a large area which will be used for display purposes only.

The next problem was the need to remember the opcodes or to use a chart and find out the opcodes of various mnemonics. Here we have eliminated the use of opcodes and instead of the opcodes the simulator provides us with the ability to enter the program directly in assembly language.

Another problem was the need to remember the sequence of instructions. The software gives a very fast solution to this problem by providing buttons to perform these functions. Here we have a button for compiling, running etc. now the user just need to click on the button and the function will be performed. This helps the user a lot as now he/she does not have to remember the sequence to be followed for running the program on the kit.

Next disadvantage is that the conversion of the code we write cannot be seen on the kit. Thus we have now developed this software on which the conversion of each statement will be shown to the user.

The kit needs explicit power supply to run where as the software will be using only the computers power to run. We do not need any explicit power supply to run the software.

Another problem we faced was the saving option. A saving option is very important for any programmer because a programmer never want to perform the work again and again. So in the software we provide the programmer with an option of saving his/her work on just a button click. The user can save his work whenever he feels like. We have provided a button on the interface which will make the user save his work very easily.

Since we have provided a memory map and we let the user see each conversion which was impossible in the kit .This makes the user to find out the errors very easily and as early as possible. So the software thus saves time by finding out errors almost instantly.

Another most important feature of the software is that the software will show the status of the flag. We have provided textboxes in which we can easily see the status of each of the flag and thus can make the decisions such as overflow condition or find out the sign status very easily by just having a look at the textboxes.

II. EXPERIMENT AND RESULT

Snapshots and Description

Main Frame

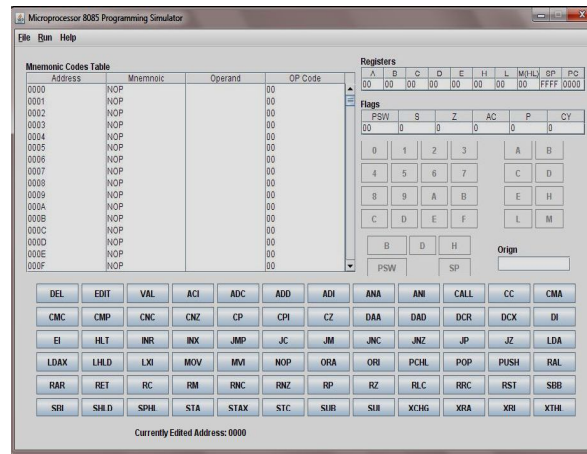


Fig 1: Main Frame

This is the main interface of simulator. This interface contains all the necessary components for an error free program to be created. On the top left there is the MNEMONIC TABLE, on the top right status of all the register is displayed in the REGISTER TABLE. Just below the register table is present the FLAG TABLE, this flag table displays the status of all the flags during the program. Below this flag table we have HEXADECIMAL NUMBER KEYS to input the address for our instruction, to it's right we have the option of choosing any of the 8 registers present in 8085, also we have the ORIGIN option to specify the address manually. And at the bottom of the interface is present all the INSTRUCTIONS that are available to the programmer, these are easy to use as need only to be clicked for initiation. Also at the bottom we can see the CURRENTLY EDITED ADDRESS.

Mnemonic Codes Table

Mnemonic Codes Table			
Address	Mnemonic	Operand	OP Code
0000	NOP		00
0001	NOP		00
0002	NOP		00
0003	NOP		00
0004	NOP		00
0005	NOP		00
0006	NOP		00
0007	NOP		00
0008	NOP		00
0009	NOP		00
000A	NOP		00
000B	NOP		00
000C	NOP		00
000D	NOP		00
000E	NOP		00
000F	NOP		00

Fig 2: Memonic Codes

This is the MNEMONIC TABLE displayed on the top left of the simulator. This table contains four columns:

- Address
- Mnemonic
- Operand
- OP Code

Address column displays the address that is being used by the programmer at that instant for execution of instructions or for storing values. Mnemonic column displays the mnemonic for the instruction being used in the program. Operand value is displayed in the operand column corresponding to the address in the address column. OP code column contains the op code, which is a one byte value that represents the corresponding instructions.

Registers Table

Registers									
A	B	C	D	E	H	L	M(HL)	SP	PC
00	00	00	00	00	00	00	00	FFFF	0000

Fig 3: Registers

This is the Register Table and provides the programmer with the content of all available registers at all the time. All the 7 registers along with memory(represented by HL), stack pointer and program counter are a part of this register table. A,B,C,D ,E,H,L all are 8 bit registers but the registers BC,DE and HL can be used in pairs to work as 16 bit storage area. Stack Pointer displays the position of the stack at that instant, and program counter displays the value of the Program Counter.

Flags Table

Flags					
PSW	S	Z	AC	P	CY
00	0	0	0	0	0

Fig 4: Flags

This is the Flag Table and displays all the five flags along with the value of the Program Status Word. Five flags being displayed here are:

- S- Sign
- Z- Zero
- AC- Auxilliary Cary
- P- Parity
- CY- Carry

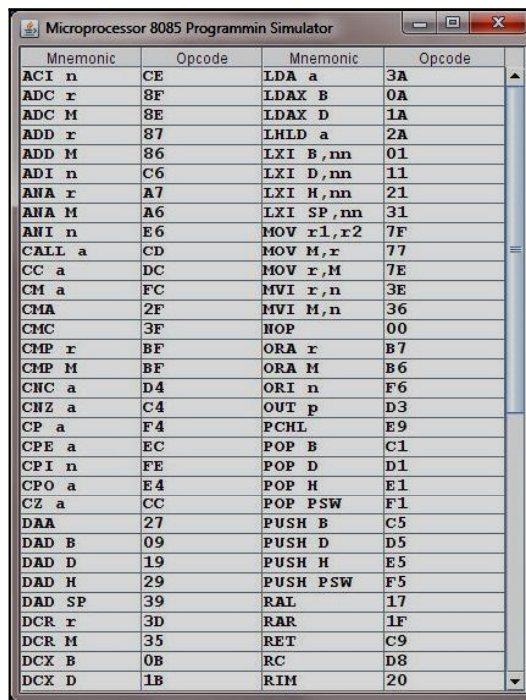
Instruction Buttons

DEL	EDIT	VAL	ACI	ADC	ADD	ADI	ANA	ANI	CALL	CC	CMA
CMC	CMP	CNC	CNZ	CP	CPI	CZ	DAA	DAD	DCR	DCX	DI
EI	HLT	INR	INX	JMP	JC	JM	JNC	JNZ	JP	JZ	LDA
LDAX	LHLD	LXI	MOV	MVI	NOP	ORA	ORI	PCHL	POP	PUSH	RAL
RAR	RET	RC	RM	RNC	RNZ	RP	RZ	RLC	RRC	RST	SBB
SBI	SBLD	SPHL	STA	STAX	STC	SUB	SUI	XCHG	XRA	XRI	XTHL

Fig 5: Instruction Buttons

This is the table that displays all the instructions that are available to the program. This table contains one address instructions, two address instructions and three address instructions. To use an instruction the programmer just needs to click the instruction, after clicking the instruction we enter the address if the instruction is 3 address example in JZ we enter the address after we click JZ. Steps to be followed after clicking the instruction is different for different instruction and depend upon the instruction itself.

Help Frame



Mnemonic	Opcode	Mnemonic	Opcode
ACI n	CE	LDA a	3A
ADC r	8F	LDAX B	0A
ADC M	8E	LDAX D	1A
ADD r	87	LHLD a	2A
ADD M	86	LXI B, nn	01
ADI n	C6	LXI D, nn	11
ANA r	A7	LXI H, nn	21
ANA M	A6	LXI SP, nn	31
ANI n	E6	MOV r1, r2	7F
CALL a	CD	MOV M, r	77
CC a	DC	MOV r, M	7E
CM a	FC	MVI r, n	3E
CMA	2F	MVI M, n	36
CMC	3F	NOP	00
CMP r	BF	ORA r	B7
CMP M	BF	ORA M	B6
CNC a	D4	ORI n	F6
CHZ a	C4	OUT p	D3
CP a	F4	PCHL	E9
CPE a	EC	POP B	C1
CPI n	FE	POP D	D1
CPO a	E4	POP H	E1
CZ a	CC	POP PSW	F1
DAA	27	PUSH B	C5
DAD B	09	PUSH D	D5
DAD D	19	PUSH H	E5
DAD H	29	PUSH PSW	F5
DAD SP	39	RAL	17
DCR r	3D	RAR	1F
DCR M	35	RET	C9
DCX B	0B	RC	D8
DCX D	1B	RIM	20

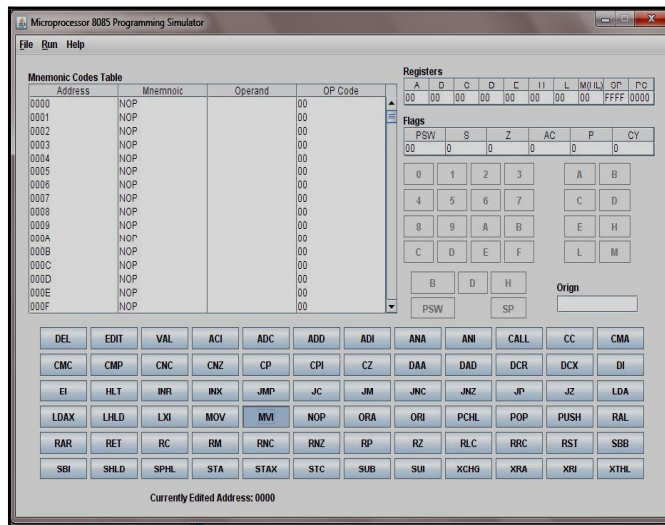
Fig 6: Help Frame

Sample Run of a program on the simulator

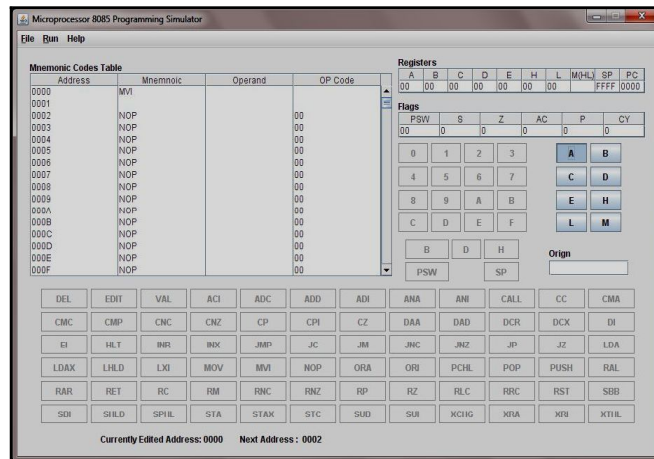
Program to move 15h to accumulator and add 9Ah to it and store the result in memory

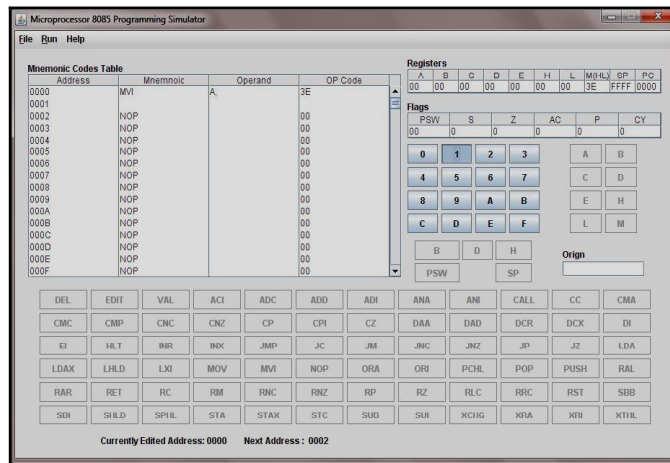
Next Pages will show how the above program can be written and saved then executed on the simulator using snapshots of the project.

Step1: Press any Instruction button to start. Example: MVI Button to Move immediate data.

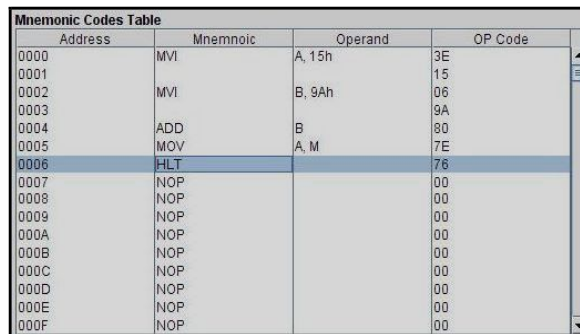


Step 2: Select the register. This selection depends upon the instruction selected in the first step.
 Example: After MOV instruction a register needs to be selected. For address instructions no register or address needs to be added.

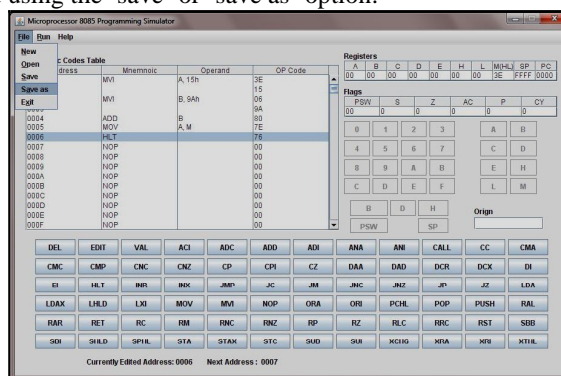




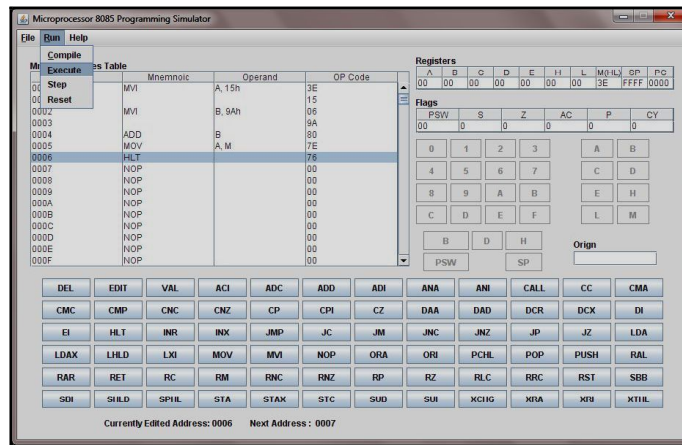
Step3: Complete the program.



Step4: Save the program if needed using the 'save' or 'save as' option.



Step 5: Execute the program using 'execute' option. Program can be compiled before execution. Program is always compiled before execution.



Step 6: Memory, registers and flags are affected according to the instructions executed.

Registers									
A	B	C	D	E	H	L	M(HL)	SP	PC
3E	9A	00	00	00	00	00	3E	FFFF	0006

Flags					
PSW	S	Z	AC	P	CY
94	1	0	1	1	0

III.CONCLUSION

FINDINGS

Finally we have explained everything about the simulator we have built right from the introduction of the project to the coding involved in it and all of us now certainly agree with the fact that the simulator has simulated the entire programming behavior of the kit which we have been using till date and has removed all the disadvantages we faced while programming with that kit

After having a look at the above stated literature survey we can say that our simulator is better than the non commercial simulators already available on the net as we have tried to overcome all the disadvantages faced in these simulators (like missing save option, not supporting subroutines, missing origin value, etc.) keeping their good features in mind and inheriting those in our simulator.

We would also like to state that the simulator implements each and every instruction with complete accuracy and also allows us to use subroutines so that we do not have to write the repeated instructions again and again, which will in turn save the memory.

IV. FUTURE SCOPE

Following additions can be made for further enhancement of the project

- A runtime graph can be included
- All the remaining instructions i.e. 5 more instructions can be implemented
- Hardware communication can be provided
- Instruction timing can be implemented

V. REFERENCE

- [1] B.Ram Principles of microprocessor, Dhanpat Rai publication
- [2] <http://www.scribd.com/doc/4616314/8085-microprocessor>
- [3] <http://www.angelfire.com/electronic2/8085Simulator/simulator.zip>
- [4] Advanced simulation topics on 8 bit microprocessor in Education conference , 1993, Twenty third annual conference, Engineering education, Renewing America's technology
- [5] A graphical simulation tool for teaching microprocessor architecture and assembly language/volume 2