

Aditya Raj 19BCS1706-- Final Practical

Algorithm:

1. Load 00 in a register C (for borrow)
2. Load two 8-bit number from memory into registers
3. Move one number to accumulator
4. Subtract the second number with accumulator
5. If borrow is not equal to 1, go to step 7
6. Increment register for borrow by 1
7. Store accumulator content in memory
8. Move content of register into accumulator
9. Store content of accumulator in other memory location
10. Stop

Code:

Memory	Mnemonics	Operands	Comment	
2000	MVI	C, 00	[C]<-00	
2002	LHLD	2500	[H-L] <- [2500]	
2005	MOV	A,H	[A] <- [H]	
2006	SUB	L	[A] <- [A] - [L]	
2007	JNC	200B	Jump if no borrow	
200A	INR	C	[C] <- [C] + 1	
200B	STA	2502	[A] -> [2502], Result	
200E	MOV	A,C	[A]<-[C]	
2010	STA	2503	[A]->[2503],Borrow	
2013	HLT		Stop	

Screenshots of simulation:

The top screenshot shows the 8085 Simulator interface. The main window is titled "8085 Simulator" and contains several tabs: Editor, Assembler, Registers, Memory, and Devices. The Assembler tab is active, displaying a table of assembly instructions:

* Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
2500		MOV A,H	7E	1	2	7
2501		INX H	23	1	1	6
2502		SBB H	9E	1	2	7
2503		INX H	23	1	1	6
2504		MOV H,A	77	1	2	7
2505		RST 1	CF	1	3	12

The Registers window on the right shows the current state of the 8085 registers:

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	00	0	0	0	0	0	0	0	0
Register L	02	0	0	0	0	0	0	1	0
Memory(H)	00	0	0	0	0	0	0	0	0

The bottom screenshot shows the 8085 Assembly Language Editor window. The main window is titled "8085 Assembly Language Editor" and contains two tabs: Assembler and Disassembler. The Assembler tab is active, displaying a list of instructions:

```
#ORG 2500H
MOV A,H
INX H
SBB H
INX H
MOV H,A
RST 1
#ORG 2501H
#DB 20,10
```

The Registers window on the right shows the current state of the 8085 registers, identical to the top screenshot.