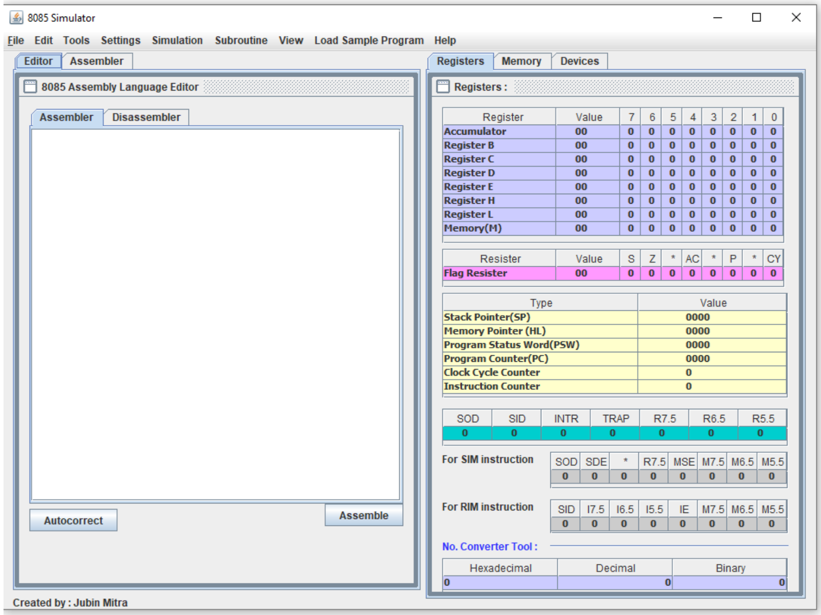
- Download 8085 compiler from the given link

  https://github.com/8085simulator/8085simulator.github.io

- **Perquisite**: You need to install **latest java version** in your PC
- In **Windows PC** simply double click **8085Compiler.jar** file. It will be open.
- In **Ubuntu PC** type this command in terminal **java -jar 8085Compiler.jar**. It will be open.
- You will see this type picture.



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	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	00	0	0	0	0	0	0	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	0000
Program Status Word(PSW)	0000
Program Counter(PC)	0000
Clock Cycle Counter	0
Instruction Counter	0

SOD	SID	INTR	TRAP	R7.5	R6.5	R5.5
0	0	0	0	0	0	0

For SIM instruction

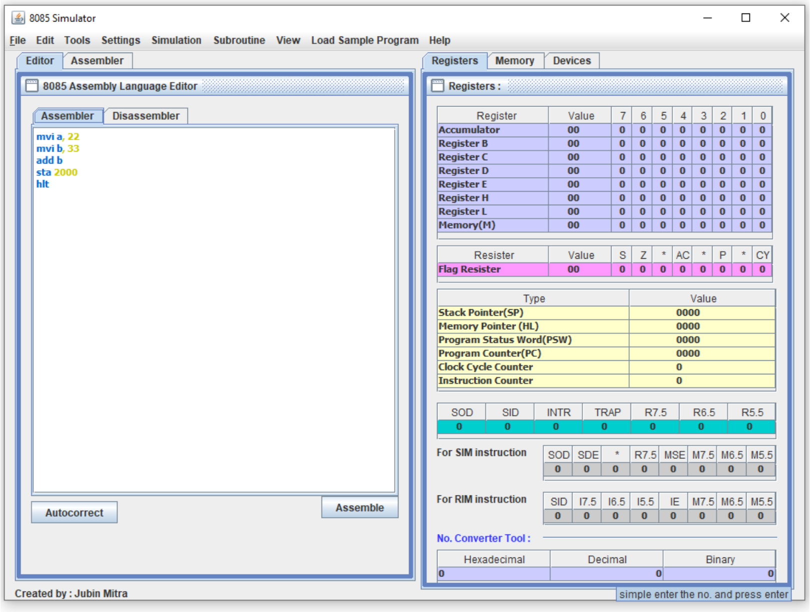
SOD	SDE	*	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

For RIM instruction

SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

Hexadecimal	Decimal	Binary
0	0	0

- You can write here your code here in assembly language.



```
mvi a, 22
mvi b, 33
add b
sta 2000
hlt
```

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	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	00	0	0	0	0	0	0	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	0000
Program Status Word(PSW)	0000
Program Counter(PC)	0000
Clock Cycle Counter	0
Instruction Counter	0

SOD	SID	INTR	TRAP	R7.5	R6.5	R5.5
0	0	0	0	0	0	0

For SIM instruction

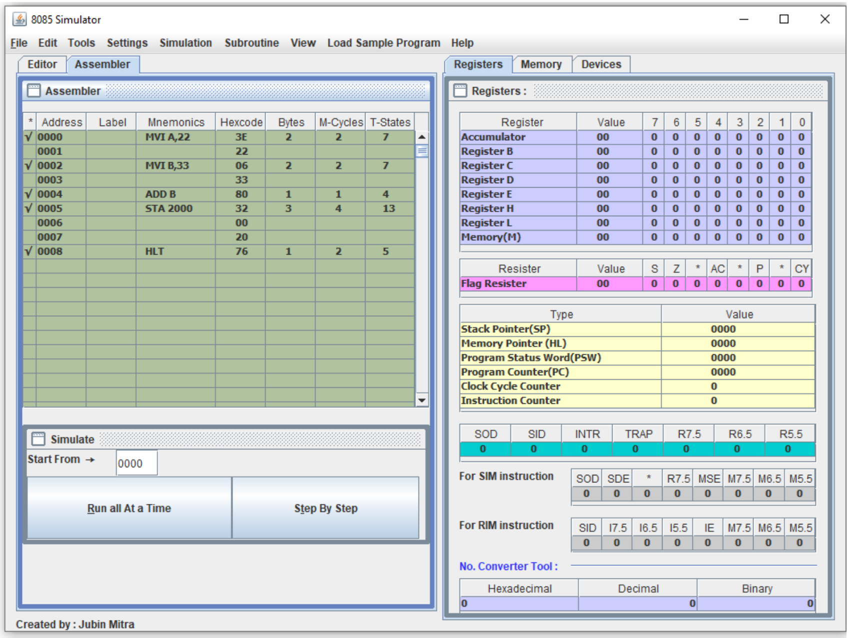
SOD	SDE	*	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

For RIM instruction

SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

Hexadecimal	Decimal	Binary
0	0	0

- After writing program you can click **Assemble** button below.



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		MVI A,22	3E	2	2	7
	0001			22			
√	0002		MVI B,33	06	2	2	7
	0003			33			
√	0004		ADD B	80	1	1	4
√	0005		STA 2000	32	3	4	13
	0006			00			
	0007			20			
√	0008		HLT	76	1	2	5

Start From → 0000

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	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	00	0	0	0	0	0	0	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	0000
Program Status Word(PSW)	0000
Program Counter(PC)	0000
Clock Cycle Counter	0
Instruction Counter	0

SOD	SID	INTR	TRAP	R7.5	R6.5	R5.5
0	0	0	0	0	0	0

For SIM instruction

SOD	SDE	*	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

For RIM instruction

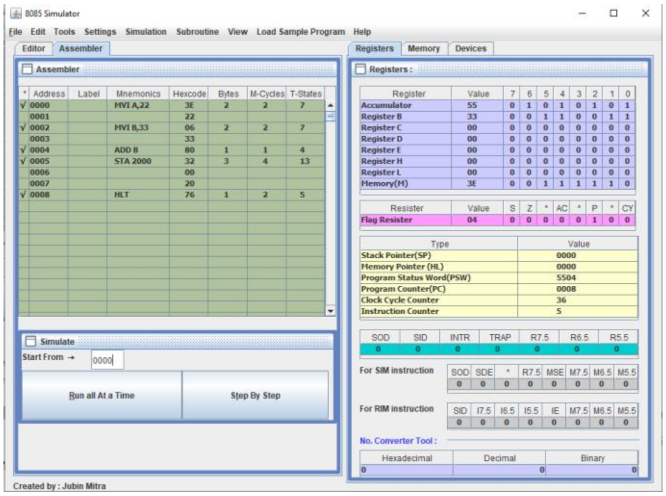
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

No. Converter Tool :

Hexadecimal	Decimal	Binary
0	0	0

Created by : Jubin Mitra

- Next you click on **Run all at time / step by step**. If you click on **Run all at time,** then it will run at once or if you click on **step by step,** then it will execute in machine cycle order. If you want to see the timing diagram of corresponding instruction, then click on corresponding instruction's **T-state** cell.



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		MVI A,22	3E	2	2	7
	0001			22			
√	0002		MVI B,33	06	2	2	7
	0003			33			
√	0004		ADD B	80	1	1	4
√	0005		STA 2000	32	3	4	13
	0006			00			
	0007			20			
√	0008		HLT	76	1	2	5

Start From → 0000

Register	Value	7	6	5	4	3	2	1	0
Accumulator	55	0	1	0	1	0	1	0	1
Register B	33	0	0	1	1	0	0	1	1
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	00	0	0	0	0	0	0	0	0
Memory(M)	3E	0	0	1	1	1	1	1	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	04	0	0	0	0	0	1	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	0000
Program Status Word(PSW)	5504
Program Counter(PC)	0008
Clock Cycle Counter	36
Instruction Counter	5

SOD	SID	INTR	TRAP	R7.5	R6.5	R5.5
0	0	0	0	0	0	0

For SIM instruction

SOD	SDE	*	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

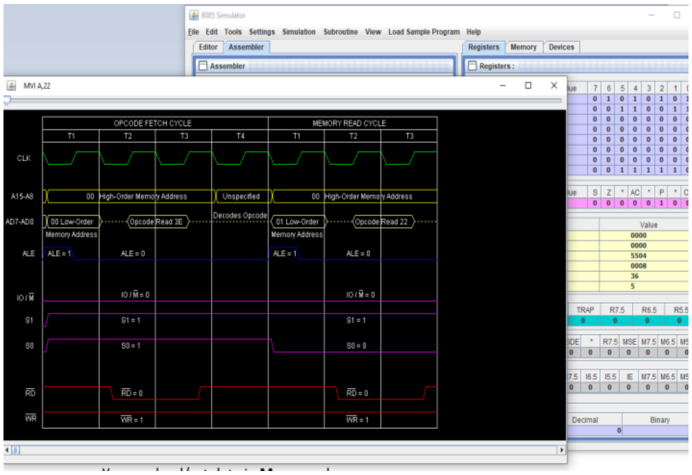
For RIM instruction

SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

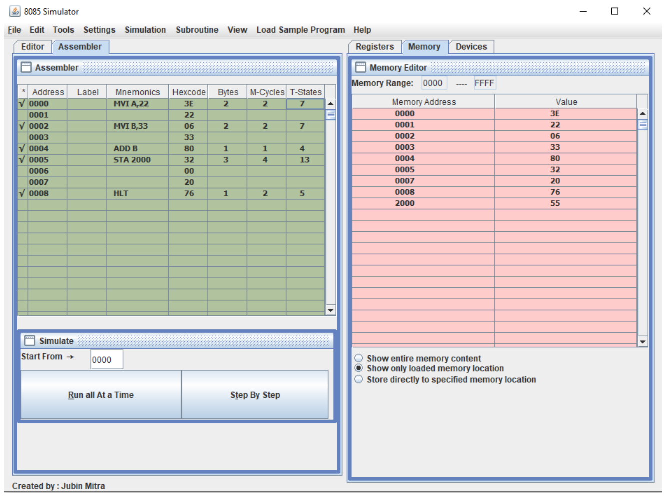
No. Converter Tool :

Hexadecimal	Decimal	Binary
0	0	0

Created by : Jubin Mitra



- You can load/set data in **Memory** also.



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		MVI A,22	3E	2	2	7
	0001			22			
√	0002		MVI B,33	06	2	2	7
	0003			33			
√	0004		ADD B	80	1	1	4
√	0005		STA 2000	32	3	4	13
	0006			00			
	0007			20			
√	0008		HLT	76	1	2	5

Start From → 0000

Memory Range: 0000 ---- FFFF

Memory Address	Value
0000	3E
0001	22
0002	06
0003	33
0004	80
0005	32
0007	20
0008	76
2000	55

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

Created by : Jubin Mitra

- In this simulator is very user friendly and very flexible.

Limitations

This or any 8085 simulator software is no way a replacement for real hardware. It only does functional simulation of the codes. It is not an emulator and hence do not expect that the timing information will be accurately modeled. However, the exact performance of the code can only be monitored in real 8085 microprocessor hardware.

Issues

Issue 1: DAA instruction wrongly toggles the carry flag if already there is a carry instead of setting it high, like take for example (88H + 88H). Users need to be cautious while using this instruction. It will be fixed in future realize v2.1.
Issue 2: In Assembler Window, during pre-processing stage of the code it flags error if ‘ ; ’ (SEMICOLON) comment marking character is followed after “ // ” (DOUBLE FORWARD SLASH).
Example! "<Label>: <Assembler Code> // <Comments> ; <More Comments>"

More details you can read the documentation uploaded in https://github.com/8085simulator/8085simulator.github.io .