

INSTITUTE OF COMPUTER TECHNOLOGY
B-TECH COMPUTER SCIENCE ENGINEERING 2025-26
SUBJECT: MICROCONTROLLER & APPLICATION

NAME: Rahul Prajapati

ENRLL. NO: 23162171020

BRANCH: CYBER SECURITY

BATCH: 52

PRACTICAL_3

Aim:- Learning Programs using Branch Instructions like JMP, JZ, JNZ, JC etc.

1. Write a Program to multiply two 8-bit numbers given as input on input ports 15H and 16H. Save the lower by of the result on memory location 3000H and higher byte of the result on 3001H.

```
MVI A,A2H // MOVE A2H IN ACCUMULATOR
OUT 15H // STORE ACCUMULATOR DATA IN MEMORY
ADDRESS 15H
MVI A,08H // MOVE 08H IN ACCUMULATOR
OUT 16H // STORE ACCUMULATOR DATA IN MEMORY
ADDRESS 15H
IN 16H // FETCH DATA FROM ADDRESS 16H
MOV C,A // MOVE ACCUMULATOR DATA INTO REGISTER
C FOR COUNTER
IN 15H // FETCH DATA FROM ADDRESS 15H
MOV D,A // MOVE ACCUMULATOR DATA INTO REGISTER
D & ITS ACT AS A MULTIPLICAND
MVI A,00H // ACCUMULATOR HOLD CURRENT SUM
MVI E,00H // ITS HOLD CARRY SUM

AGAIN: ADD D // AGAIN IS LABEL WHICH IS ACT AS A LOOP
AND ADD D INSTRUCTION IN ADD DATA D INTO REGISTER A
JNC CARRY // IT WILL JUMP IF THERE WILL
BE NO CARRY
IN R E // INCREMENT HIGHER BYTE

CARRY: DCR C // DECREMENT COUNTER
JNZ AGAIN // IT WILL JUMP ON AGAIN WHEN
ZERO FLAG IS RESET
STA 3000H // STORE LOWERBYTE RESULT AT 3000H
MOV A,E // MOVE HIGHER BYTE IN ACCUMULATOR
STA 3001H // STORE HIGHER BYTE AT 3001H
HLT // HOLD
```

Editor Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
✓	000A		MOV C,A	4F	1	1	4
✓	000B		IN 15	DB	2	3	10
	000C			15			
✓	000D		MOV D,A	57	1	1	4
✓	000E		MVI A,00	3E	2	2	7
	000F			00			
✓	0010		MVI E,00	1E	2	2	7
	0011			00			
✓	0012	AGAIN	ADD D	82	1	1	4
✓	0013		JNC CARRY	D2	3	3	10
	0014			17			
	0015			00			
✓	0016		INR E	1C	1	1	4
✓	0017	CARRY	DCR C	0D	1	1	4
✓	0018		JNZ AGAIN	C2	3	3	10
	0019			12			
	001A			00			
✓	001B		STA 3000	32	3	4	13
	001C			00			
	001D			00			

HOLD

Simulate

Start From →

Run all At a Time	Step By Step
-------------------	--------------

Registers Memory Devices

Registers :

Register	Value	7	6	5	4	3	2	1	0
Accumulator	05	0	0	0	0	0	1	0	1
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	A2	1	0	1	0	0	0	1	0
Register E	05	0	0	0	0	0	1	0	1
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 Register	55	0	1	0	1	0	1	0	1

Type	Value							
Stack Pointer(SP)	0000							
Memory Pointer (HL)	0000							
Program Status Word(PSW)	0555							
Program Counter(PC)	0022							
Clock Cycle Counter	342							
Instruction Counter	52							

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	0	0	0

0022	76
3000	10
3001	05

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

ENG IN 19:00 18-08-2025

2. Write a Program to count positive and negative numbers out of 20 numbers stored In memory. (Assume appropriate memory location in your program and load 20 different bytes in memory using assembler directives). Display the count of positive nos on output port 30 H and negative numbers on 40 H.

Assembler	Disassembler
# ORG 5000H # DB F1H,52H,28H,B1H,A9H,08H,13H,36H,26H,88H,FFH,E4H,22H,31H,65H,44H # ORG 5100H	LXI H,5000H // LOAD THE ADDRESS 5000H INTO HL PAIR MVI C,14H // STORE 14H IN REGISTER C MVI D,00H // STORE 00H IN REGISTER D FOR //POSITIVE COUNT MVI E,00H // STORE 14H IN REGISTER E FOR //NEGATIVE COUNT NEXT: MOV A,M // MOVE MEMORY DATA INTO ACCUMULATOR ANI 80H // PERFORM AND OPERATION TO //IDENTIFY MSB 1 OR 0 JZ POSITIVE // IF SIGN BIT IS 0 THEN JUMP ON // POSITIVE NEG: INR E // INCREMENT NEGATIVE COUNTER JMP CONTINUE // JUMP ON CONTINUE POSITIVE: INR D // INCREMENT POSITIVE COUNTER CONTINUE: INX H // INCREMENT HL PAIR DCR C // DECREMENT MAIN COUNTER JNZ NEXT // JUMP ON NEXT IF ZERO FLAG IS NOT // SET TO 0 MOV A,D // MOVE POSITIVE COUNTS TO //ACCUMULATOR OUT 30H // STORE ACCUMULATOR DATA INTO // MEMORY ADDRESS 30H MOV A,E // MOVE NEGATIVE COUNTS TO ACCUMULATOR OUT 40H // STORE ACCUMULATOR DATA INTO // MEMORY ADDRESS 40H

Assembler							
*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
	5115			14			
✓	5116		MVI D,00	16	2	2	7
	5117			00			
✓	5118		MVI E,00	1E	2	2	7
	5119			00			
✓	511A	NEXT	MOV A,M	7E	1	2	7
✓	511B		ANI 80	E6	2	2	7
	511C			80			
✓	511D		JZ POSITIVE	CA	3	3	10
	511E			24			
	511F			51			
✓	5120	NEG	INR E	1C	1	1	4
✓	5121		JMP CONTI...	C3	3	3	10
	5122			25			
	5123			51			
✓	5124	POSIT...	INR D	14	1	1	4
	5125	CONT...	INX H	23	1	1	6
	5126		DCR C	0D	1	1	4
	5127		INZ NEXT	C2	3	3	10

IF SIGN BIT IS 0 THEN JUMP ON

 Simulate	
Start From →	<input type="text" value="5111"/>
Run all At A Time	Step By Step

Registers		Memory		Devices						
Registers :										
Register	Value	7	6	5	4	3	2	1	0	
Accumulator	06	0	0	0	0	0	1	1	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	0E	0	0	0	0	1	1	1	0	
Register E	06	0	0	0	0	0	1	1	0	
Register H	50	0	1	0	1	0	0	0	0	
Register L	14	0	0	0	1	0	1	0	0	
Memory(M)	00	0	0	0	0	0	0	0	0	
Register	Value	S	Z	*	AC	*	P	*	CY	
Flag Register	54	0	1	0	1	0	1	0	0	
Type					Value					
Stack Pointer(SP)					0000					
Memory Pointer (HL)					5014					
Program Status Word(PSW)					0654					
Program Counter(PC)					5130					
Clock Cycle Counter					1568					
Instruction Counter					228					
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				

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	0E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	06	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

3. Write a program to add 10 bytes stored in a string starting from 1000H .Store result at 2000H (LSB), 2001H (MSB).

Assembler		Disassembler	
# ORG 1000H # DB 05H,12H,23H,34H,45H,56H,67H,78H,89H,9AH # ORG 1100H			
LXI H,1000H // LOAD FIRST DATA'S ADDRESS			
INTO HL PAIR			
MVI C,0AH // COUNTER			
MVI D,00H // REGISTER FOR HIGHER BYTE			
MVI E,00H // REGISTER FOR LOWER BYTE			
NEXT: MOV A,M // FETCH DATA FROM MEMORY ADDRESS AND STORE IT TO ACCUMULATOR			
ADD E // ADD LSB			
MOV E,A // STORE LSB IN REGISTER E			
JNC LABEL // JUMP ON LABEL IF CARRY FLAG IS SET TO 0			
IN R D // INCREMENT MSB			
LABEL: INX H // INCREMENT HL PAIR TO GET NEXT DATA			
DCR C // DECREMENT COUNTER			
JNZ NEXT // JUMP ON NEXT IF ZERO FLAG IS SET TO 0			
LXI H,2000H // LOAD 2000H ADDRESS IN HL			
PAIR			
MOV M,E // STORE LSB IN MEMORY LOCATION WHICH IS DEFINED IN HL PAIR			
INX H // INCREMENT HL PAIR			
MOV M,D // STORE MSB IN MEMORY LOCATION WHICH IS DEFINED IN HL PAIR			
HLT // stop			

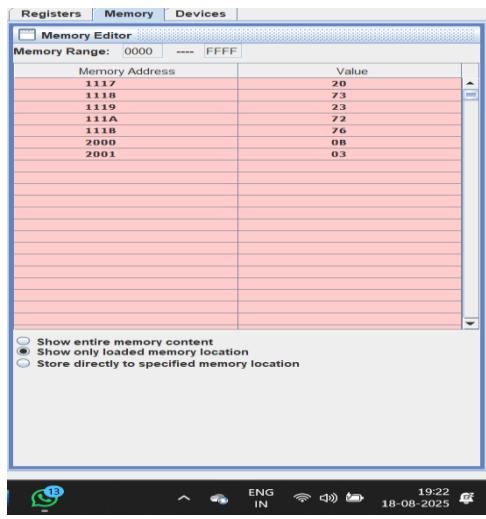
Editor		Assembler	
<input type="checkbox"/> Assembler		<input type="checkbox"/> Registers	
*	Address	Label	Mnemonics
✓	1100		LXI H,1000
			21
			3
			3
			10
✓	1101		
			00
			10
✓	1102		
			0A
✓	1103		MVI C,0A
			0E
			2
			2
			7
✓	1104		
			0A
✓	1105		MVI D,00
			16
			2
			2
			7
✓	1106		
			00
✓	1107		MVI E,00
			1E
			2
			2
			7
✓	1108		
			00
✓	1109	NEXT	MOV A,M
			7E
			1
			2
			7
✓	110A		ADD E
			83
			1
			4
✓	110B		MOV E,A
			5F
			1
			4
✓	110C		JNC LABEL
			D2
			3
			3
			10
✓	110D		
			10
✓	110E		
			11
✓	110F		INR D
			14
			1
			1
			4
✓	1110	LABEL	INX H
			23
			1
			1
			6
✓	1111		DCR C
			0D
			1
			1
			4
✓	1112		JNZ NEXT
			C2
			3
			3
			10

Simulate	
Start From →	1000
Run all At a Time	Step By Step

Registers	
Register	Value
Accumulator	0B
Register B	01
Register C	00
Register D	03
Register E	0B
Register H	20
Register L	01
Memory(M)	03
Register	Value
Flag Register	S Z AC P CY
Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	2001
Program Status Word(PSW)	0B55
Program Counter(PC)	111B
Clock Cycle Counter	180682
Instruction Counter	45120
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
For RIM instruction SID I7.5 I6.5 I5.5 IE M7.5 M6.5 M5.5	0 0 0 0 0 0 0
No. Converter Tool : Hexadecimal Decimal Binary	0 0

Created by : Jubin Mitra

ENG IN 19:21 18-08-2025



4. Write a program to subtract the contents of register H from register L without using any of subtract instruction.

Assembler		Disassembler	
<pre>MVI H,05H // STORE 05H IN REGISTER H MVI L,0AH // STORE 0AH IN REGISTER L // Program to subtract H from L using 2's complement MOV A,H // MOVE REGITER H DATA INTO ACCUMULATOR CMA // PERFORM 1'S COMPLIMENT OF ACCUMULATOR INR A // ADD 1 IN ACCUMULATOR DATA FOR 2'S COMPLIMENT ADD L // ADD L INTO H THEN SUBTRACTION PERFORM MOV L,A // STORE RESULT IN REGISTER L HLT // STOP // Program to subtract H from L using 1's complement MOV A,H // MOVE REGITER H DATA INTO ACCUMULATOR CMA // PERFORM 1'S COMPLIMENT OF ACCUMULATOR ADD L // PERFORM ADD OPERATION WITH REGISTER A AND L INR A // ADD 1 IN ACCUMULATOR'S DATA MOV L,A // STORE RESULT IN REGITER L HLT // STOP</pre>			

(a) Apply 2's Complement method.

Editor		Assembler		Registers		Memory		Devices																																																																																																																																																																																																																					
<table border="1"> <thead> <tr> <th>*</th> <th>Address</th> <th>Label</th> <th>Mnemonics</th> <th>Hexcode</th> <th>Bytes</th> <th>M-Cycles</th> <th>T-States</th> </tr> </thead> <tbody> <tr><td>✓</td><td>0000</td><td></td><td>MVI H,05</td><td>26</td><td>2</td><td>2</td><td>7</td></tr> <tr><td>✓</td><td>0001</td><td></td><td></td><td>05</td><td></td><td></td><td></td></tr> <tr><td>✓</td><td>0002</td><td></td><td>MVI L,0A</td><td>2E</td><td>2</td><td>2</td><td>7</td></tr> <tr><td>✓</td><td>0003</td><td></td><td></td><td>0A</td><td></td><td></td><td></td></tr> <tr><td>✓</td><td>0004</td><td></td><td>MOV A,H</td><td>7C</td><td>1</td><td>1</td><td>4</td></tr> <tr><td>✓</td><td>0005</td><td></td><td>CMA</td><td>2F</td><td>1</td><td>1</td><td>4</td></tr> <tr><td>✓</td><td>0006</td><td></td><td>INR A</td><td>3C</td><td>1</td><td>1</td><td>4</td></tr> <tr><td>✓</td><td>0007</td><td></td><td>ADD L</td><td>85</td><td>1</td><td>1</td><td>4</td></tr> <tr><td>✓</td><td>0008</td><td></td><td>MOV L,A</td><td>6F</td><td>1</td><td>1</td><td>4</td></tr> <tr><td>✓</td><td>0009</td><td></td><td>HLT</td><td>76</td><td>1</td><td>2</td><td>5</td></tr> </tbody> </table>		*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States	✓	0000		MVI H,05	26	2	2	7	✓	0001			05				✓	0002		MVI L,0A	2E	2	2	7	✓	0003			0A				✓	0004		MOV A,H	7C	1	1	4	✓	0005		CMA	2F	1	1	4	✓	0006		INR A	3C	1	1	4	✓	0007		ADD L	85	1	1	4	✓	0008		MOV L,A	6F	1	1	4	✓	0009		HLT	76	1	2	5	<table border="1"> <thead> <tr> <th colspan="2">Simulate</th> </tr> <tr> <th>Start From</th> <th>→ 0000</th> </tr> </thead> <tbody> <tr> <td colspan="2"> <input type="button" value="Run all At a Time"/> <input type="button" value="Step By Step"/> </td> </tr> </tbody> </table>		Simulate		Start From	→ 0000	<input type="button" value="Run all At a Time"/> <input type="button" value="Step By Step"/>		<table border="1"> <thead> <tr> <th>Register</th> <th>Value</th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr><td>Accumulator</td><td>05</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr> <tr><td>Register B</td><td>00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>Register C</td><td>00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>Register D</td><td>00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>Register E</td><td>00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>Register H</td><td>05</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>Register L</td><td>05</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>Memory(M)</td><td>00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </tbody> </table>		Register	Value	7	6	5	4	3	2	1	0	Accumulator	05	0	0	0	0	1	0	1		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	05	0	0	0	0	0	1	0	1	Register L	05	0	0	0	0	0	1	0	1	Memory(M)	00	0	0	0	0	0	0	0	0	<table border="1"> <thead> <tr> <th>Register</th> <th>Value</th> <th>S</th> <th>Z</th> <th>* AC</th> <th>* P</th> <th>* CY</th> </tr> </thead> <tbody> <tr><td>Flag Register</td><td>15</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> </tbody> </table>		Register	Value	S	Z	* AC	* P	* CY	Flag Register	15	0	0	1	0	1	<table border="1"> <thead> <tr> <th>Type</th> <th>Value</th> </tr> </thead> <tbody> <tr><td>Stack Pointer(SP)</td><td>0000</td></tr> <tr><td>Memory Pointer (HL)</td><td>0505</td></tr> <tr><td>Program Status Word(PSW)</td><td>0515</td></tr> <tr><td>Program Counter(PC)</td><td>0009</td></tr> <tr><td>Clock Cycle Counter</td><td>39</td></tr> <tr><td>Instruction Counter</td><td>8</td></tr> </tbody> </table>		Type	Value	Stack Pointer(SP)	0000	Memory Pointer (HL)	0505	Program Status Word(PSW)	0515	Program Counter(PC)	0009	Clock Cycle Counter	39	Instruction Counter	8
*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States																																																																																																																																																																																																																						
✓	0000		MVI H,05	26	2	2	7																																																																																																																																																																																																																						
✓	0001			05																																																																																																																																																																																																																									
✓	0002		MVI L,0A	2E	2	2	7																																																																																																																																																																																																																						
✓	0003			0A																																																																																																																																																																																																																									
✓	0004		MOV A,H	7C	1	1	4																																																																																																																																																																																																																						
✓	0005		CMA	2F	1	1	4																																																																																																																																																																																																																						
✓	0006		INR A	3C	1	1	4																																																																																																																																																																																																																						
✓	0007		ADD L	85	1	1	4																																																																																																																																																																																																																						
✓	0008		MOV L,A	6F	1	1	4																																																																																																																																																																																																																						
✓	0009		HLT	76	1	2	5																																																																																																																																																																																																																						
Simulate																																																																																																																																																																																																																													
Start From	→ 0000																																																																																																																																																																																																																												
<input type="button" value="Run all At a Time"/> <input type="button" value="Step By Step"/>																																																																																																																																																																																																																													
Register	Value	7	6	5	4	3	2	1	0																																																																																																																																																																																																																				
Accumulator	05	0	0	0	0	1	0	1																																																																																																																																																																																																																					
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	05	0	0	0	0	0	1	0	1																																																																																																																																																																																																																				
Register L	05	0	0	0	0	0	1	0	1																																																																																																																																																																																																																				
Memory(M)	00	0	0	0	0	0	0	0	0																																																																																																																																																																																																																				
Register	Value	S	Z	* AC	* P	* CY																																																																																																																																																																																																																							
Flag Register	15	0	0	1	0	1																																																																																																																																																																																																																							
Type	Value																																																																																																																																																																																																																												
Stack Pointer(SP)	0000																																																																																																																																																																																																																												
Memory Pointer (HL)	0505																																																																																																																																																																																																																												
Program Status Word(PSW)	0515																																																																																																																																																																																																																												
Program Counter(PC)	0009																																																																																																																																																																																																																												
Clock Cycle Counter	39																																																																																																																																																																																																																												
Instruction Counter	8																																																																																																																																																																																																																												
<table border="1"> <thead> <tr> <th>SOD</th> <th>SID</th> <th>INTR</th> <th>TRAP</th> <th>R7.5</th> <th>R8.5</th> <th>R5.5</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </tbody> </table>		SOD	SID	INTR	TRAP	R7.5	R8.5	R5.5	0	0	0	0	0	0	0	<table border="1"> <thead> <tr> <th>For SIM instruction</th> <th>SOD</th> <th>SDE</th> <th>*</th> <th>R7.5</th> <th>MSE</th> <th>M7.5</th> <th>M6.5</th> <th>M5.5</th> </tr> </thead> <tbody> <tr><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </tbody> </table>		For SIM instruction	SOD	SDE	*	R7.5	MSE	M7.5	M6.5	M5.5		0	0	0	0	0	0	0	0	<table border="1"> <thead> <tr> <th>For RIM instruction</th> <th>SID</th> <th>I7.5</th> <th>I6.5</th> <th>I5.5</th> <th>IE</th> <th>M7.5</th> <th>M6.5</th> <th>M5.5</th> </tr> </thead> <tbody> <tr><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </tbody> </table>		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	<table border="1"> <thead> <tr> <th>No. Converter Tool :</th> </tr> </thead> <tbody> <tr> <td>Hexadecimal</td> <td>Decimal</td> <td>Binary</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>		No. Converter Tool :	Hexadecimal	Decimal	Binary	0	0	0																																																																																																																																																													
SOD	SID	INTR	TRAP	R7.5	R8.5	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



(b) Apply 1's Complement method.

The screenshot shows a CPU simulation interface with the following components:

- Assembler Window:** Displays assembly code from address 0000 to 0009. The code includes instructions like MVI H,05, MVI L,0A, MOV A,H, CMA, ADD L, INR A, MOV I,A, and HLT.
- Registers Window:** Shows the state of various registers. Register L is highlighted in red, indicating it is the current register being viewed. The Flag Register shows S=0, Z=0, AC=0, P=1, CY=1.
- Memory Window:** Shows the memory starting at address 0000 with value 05.
- Simulate Window:** Contains buttons for "Start From" (set to 0000), "Run all At a Time", and "Step By Step".
- System Status Bar:** Includes icons for system status, language (ENG IN), date (18-08-2025), and time (19:32).