GYAN GANGA INSTITUTE OF TECHNOLOGY AND SCIENCES, JABALPUR

CSE

LAB MANUAL

Computer Org. & Architecture

Computer Science Engineering

NAME:

_____

ENROLLMENT NUMBER:

_____

INDEX

Sr.No.	Object	Date	signature	Remark
1.	Study of Multiplexer and De multiplexer			
2	Study of Half Adder and Subtractor			
3.	Study of Full Adder and Subtractor			
4.	WAP to find the 2's compliment of a given number.			
5.	WAP to transfer 5(51,52,53,54&55) numbers in to five successive memory location starting from 2000h.			
6.	WAP to add two 8-bit numbers and store the result at memory location2000h			
7.	WAP to add two 16-bit numbers. Store the result at memory address starting from 2000.			
8.	WAP to subtract two 8-bit numbers and store the result at memory location 2000.			
9	WAP to multiply two 8-bit numbers stored at memory location 2000h and 2001h and stores the result at memory location 2000 h and 2001h			
10.	Assume that 3 bytes of data are stored at consecutive memory addresses of the data memory starting at 2000. Write a program which Loads register C with (2000), i.e. with data contained at memory address2000, D with (2001), E with (2002) and A with (2001).			

11.	**Sixteen bytes of data are specified at consecutive data-memory locations starting at 2000. Write a program which increments the value of all sixteen bytes by01.**			
12.	**WAP to add t 10 bytes stored at memory location starting from 3000. Store the result at memory location300A**			

Gyan Ganga Institute of Technology and Sciences, Jabalpur

Computer Science Engineering

Practical No. 1

Aim: Construction and verification of Multiplexer and Demultiplexer

Software:

MULTISIM

Theory:

Multiplexer

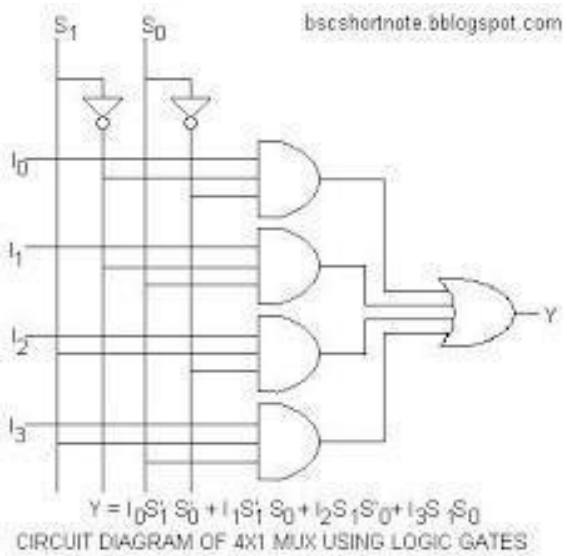
A multiplexer is a circuit which has a number of inputs but only 1 output or we can say multiplexer is a circuit which transmits a large number of information signals(inputs) over a small number of signal lines(output). Digital multiplexer is a combinational logic circuit and its function is to select information in binary from one of many inputs and outputs the information along a single selected output. These circuits are especially useful when a complex logic circuit is to be shared by a number of input signals. The information to be outputted is selected by the address line.

In case of 4:1 multiplexer, it has four input lines having a signal as I0, I1, I2 AND I3. For selecting one of the four input signals we require address which can be a two-bit word. The address lines are designated as S1 and S2. For each combination of selection signals (S1 and S0) one of the inputs is outputted.

Truth Table:

S. No	SELECTION INPUT		OUTPUT
	S1	S2	Y
1.	0	0	I_0
2.	0	1	I_1
3.	1	0	I_2
4.	1	1	I_3

Circuit Diagram:



CIRCUIT DIAGRAM OF 4X1 MUX USING LOGIC GATES

Demultiplexer

Demultiplexer as the name indicates, it has only data input D with 4 outputs namely Q0, Q1, Q2, Q3 It has two data selector inputs namely S0, S1, at which control bits are applied.

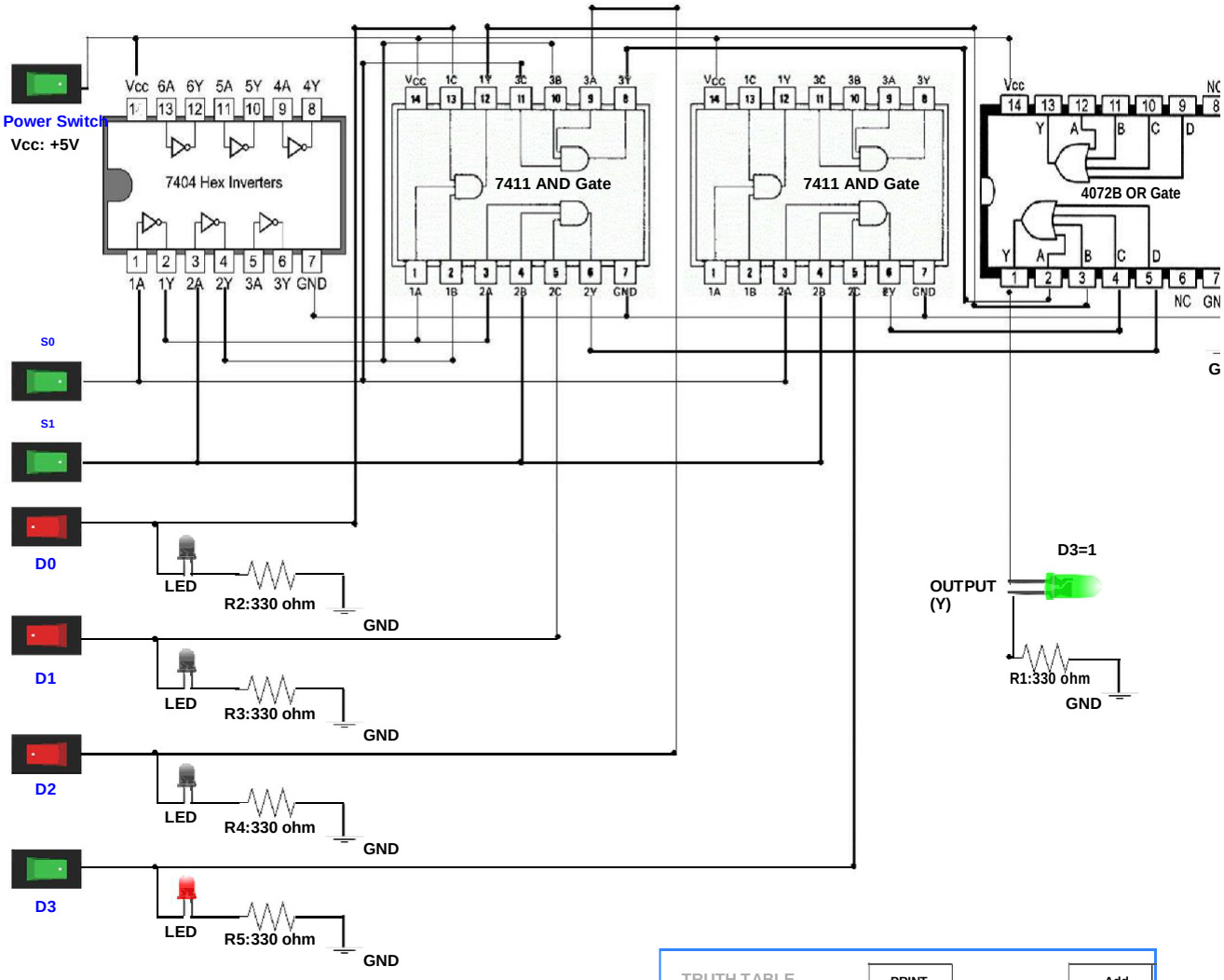
The data bit is transmitted to the data bit Q0, Q1, Q2, Q3 of the output lines. Which particular output line will be chosen will depend on the value of S1, S0 the control input. Consider the case when S1 S0=00 now the upper AND gate is enabled while all other AND gate are disabled. Hence it is not possible to activate any output other than Y0. Thus Q0=D, if D is low Q0 will be low and if D is high, Q0 will be high. Considering another case, S1, S0=01. We find that Q1 is activated because second AND gate is enabled. Similarly, if S1 S0=11.

Truth Table:

Data Input	Select Inputs		Outputs			
D_{in}	S1	S0	Q3	Q2	Q1	Q0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

INSTRUCTIONS

Experiment to perform logic of 4:1 Multiplexer on kit



TRUTH TABLE PRINT Add

Serial No.	S0	S1	OUTPUT (Y)	OUTPUT VALUE
1	0	0	D0	0
2	0	1	D1	0
3	1	0	D2	0
4	1	1	D3	0

Gyan Ganga Institute of Technology and Sciences, Jabalpur

AComputer Science Engineering

Practical No. 2 Aim:

Construction and verification of Half Adder and Subtractor.

Virtual Lab **Software:**

Theory:

Half Adder

It is combinational circuits that perform addition of two bits. This circuit has two inputs A and B (augend and added) and two outputs- Sum (S) and Carry (C). The sum is a 1 when A and B are different and carry is a 1when A and B are 1. The truth table for a half adder can be constructed using the addition table for binary numbers

Truth Table

INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the truth table, we can write logical expression for S and C outputs as

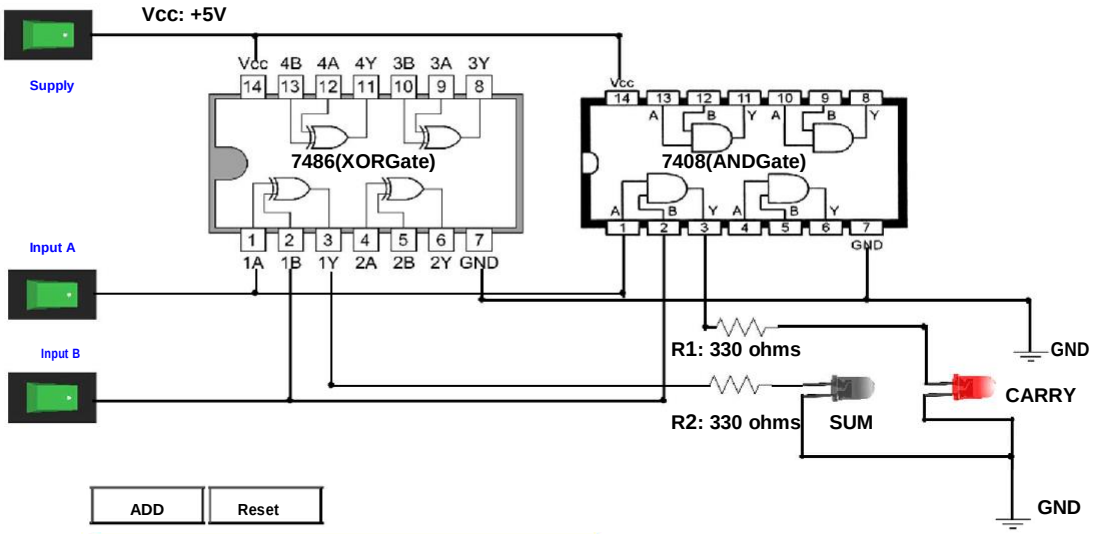
$$S = AB + \overline{A}B = A \text{ exor } B$$

$$C=AB$$

From the equation it is clear that this 1-bit adder can be easily implemented with the help of EXOR Gate for the output 'SUM' and an AND Gate for the carry. Take a look at the implementation below.

INSTRUCTIONS

Experiment to perform logic of half adder on kit



TRUTH TABLE

Print

Serial No.	Input A	Input B	Sum	Carry
1	0	0	0	0
2	0	1	1	0
3	1	0	1	0
4	1	1	0	1

Fig:1 Logic Circuit Diagram of Implementation of Half Adder

Half Subtractor:

A combinational circuit which performs the subtraction of two bits is called half subtractor. The input variables designate the minuend and the subtrahend bit, whereas the output variables produce the difference and borrow bits.

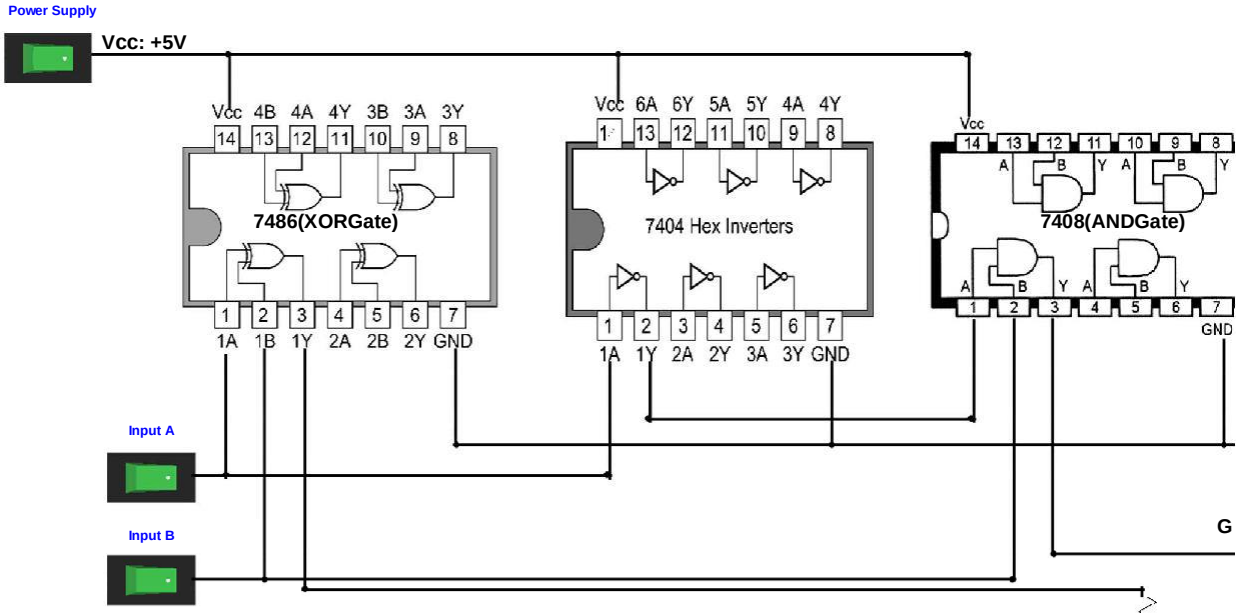
The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow).

INPUTS		OUTPUTS	
A	B	Diff	Borrow
0	0	0	0
1	1	1	0
1	0	1	0
1	1	0	0

Result: - Thus, we have verified the truth table of Half Adder and Full Subtractor on

INSTRUCTIONS

Experiment to perform logic of half Subtractor on kit



TRUTH TABLE

ADD PRINT

Serial No.	Input A	Input B	Difference	Borrow
1	0	0	0	0
2	0	1	1	1
3	1	0	1	0
4	1	1	0	0

Gyan Ganga Institute of Technology and Sciences, Jabalpur

Computer Science Engineering

Practical No. 3

Aim:

Construction and verification Full Adder and Subtractor

Software: **VIRTUAL LAB**

Theory: **Full Adder**

This type of adder is a little more difficult to implement than a half-adder. The main difference between a half-adder and a full-adder is that the full-adder has three inputs and two outputs. The first two inputs are A and B and the third input is an input carry designated as CIN. When a full adder logic is designed, we will be able to string eight of them together to create a byte-wide adder and cascade the carry bit from one adder to the next.

The output carry is designated as COUT and the normal output is designated as S. Take a look at the truth-table.

	INPUTS		OUTPUTS	
A	B	C_{IN}	COUT	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

From the above truth-table, the full adder logic can be implemented. We can see that the output S is an EXOR between the input A and the half-adder SUM output with B and CIN inputs. We must also note that the COUT will only be true if any of the two inputs out of the three are HIGH.

$$S=A \text{ EXOR } B \text{ EXOR }$$

$$Cout = AB + ACin + BCin$$

Thus, we can implement a full adder circuit with the help of two half adder circuits. The first will half adder will be used to add A and B to produce a partial Sum. The second half adder logic can be used to add CIN to the Sum produced by the first half adder to get the final S output. If any of the half adder logic produces a carry, there will be an output carry. Thus, COUT will be an OR function of the half-adder Carry outputs.

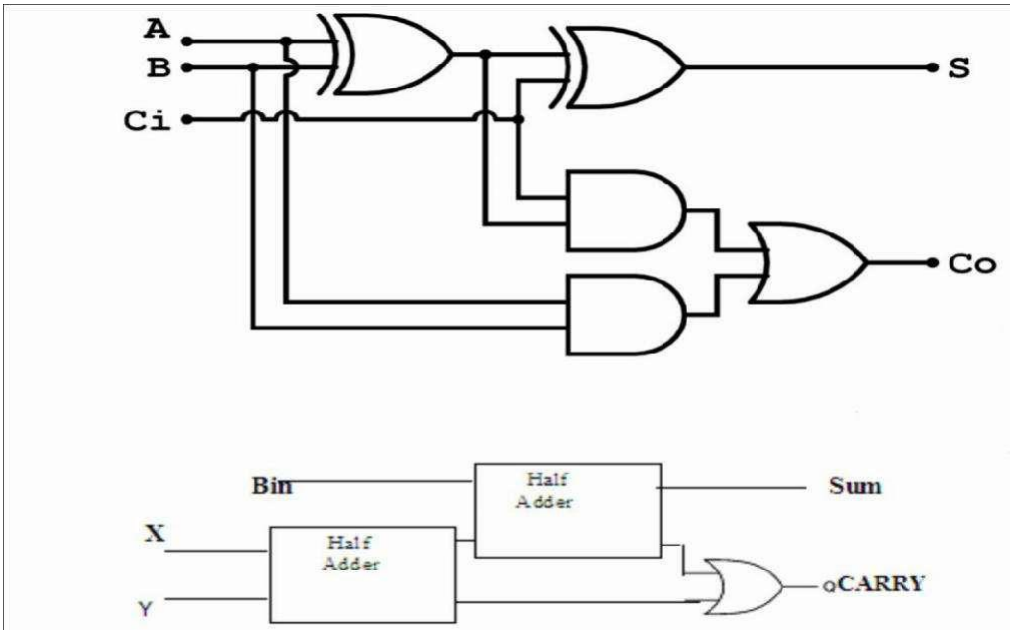


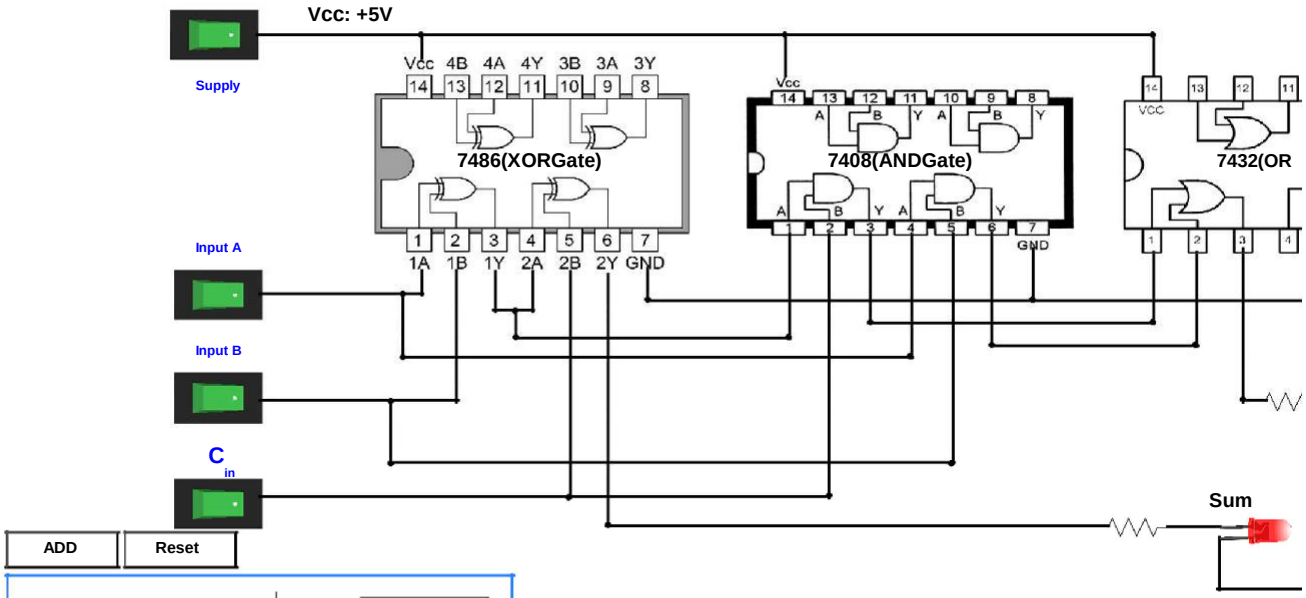
Fig 2 Full Adder using Half Adder.

Single Bit full Adder

With this type of symbol, we can add two bits together taking a carry from the next lower order of magnitude, and sending a carry to the next higher order of magnitude. In a computer, for a multi-bit operation, each bit must be represented by a full adder and must be added simultaneously. Thus, to add two 8-bit numbers, you will need 8 full adders which can be formed by cascading two of the 4-bit blocks. The addition of two 4-bit numbers is shown below.

INSTRUCTIONS

Experiment to perform logic of Full Adder on kit



TRUTH TABLE

Print

Serial No.	Input A	Input B	C_{in}	Carry OUT	Sum
1	0	0	0	0	0
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	1	0
5	1	0	0	0	1
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	1	1

Full Subtractor

A full subtractor is a combinational circuit that performs subtraction involving three bits, namely A (minuend), B (subtrahend), and Bin (borrow-in). It accepts three inputs: A (minuend), B (subtrahend) and a Bin (borrow bit) and it produces two outputs: D (difference) and Bout (borrow out). The logic symbol and truth table are shown below.

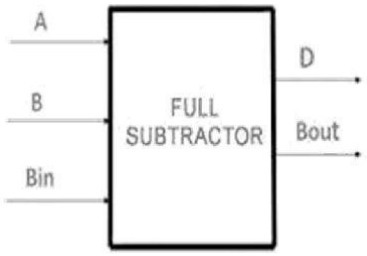


Figure-3: Logic Symbol of Full subtractor

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figure-4: Truth Table of Full subtractor

From the above truth table, we can find the Boolean expression.

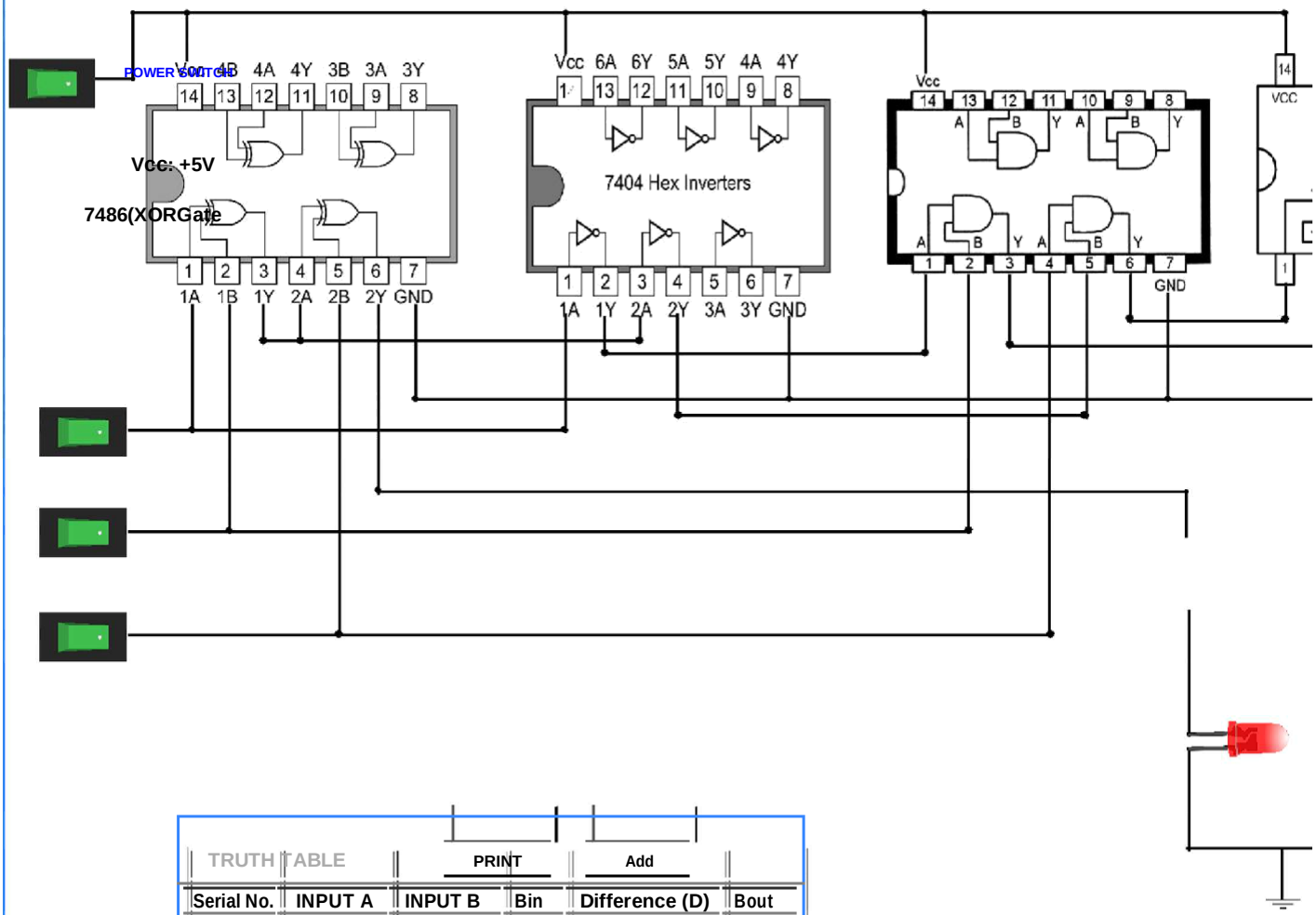
$$D = A \oplus B \oplus Bin$$

$$Bout = A' Bin + A' B + B Bin$$

From the equation we can draw the Full-subtractor circuit as shown in the figure 6.

Result: - **Thus, we have verified the truth table of Full Adder and Subtractor on MUTISIM.**

Experiment to perform logic of Full Subtractor on kit



TRUTH TABLE		PRINT	Add		
Serial No.	INPUT A	INPUT B	Bin	Difference (D)	Bout
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	0	1	1
4	0	1	1	0	1
5	1	0	0	1	0
6	1	0	1	0	0
7	1	1	1	1	1

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science Engineering

Practical No. 4

Aim:

WAP to find the 2's compliment of a given number.

Software:

GNUSIM-8085

Theory

To determine the complement of a given number to the number into accumulator then compliment all the bits of the number after complementing all the bits at 012 this number and the 2's complement will be achieved

Algorithm –

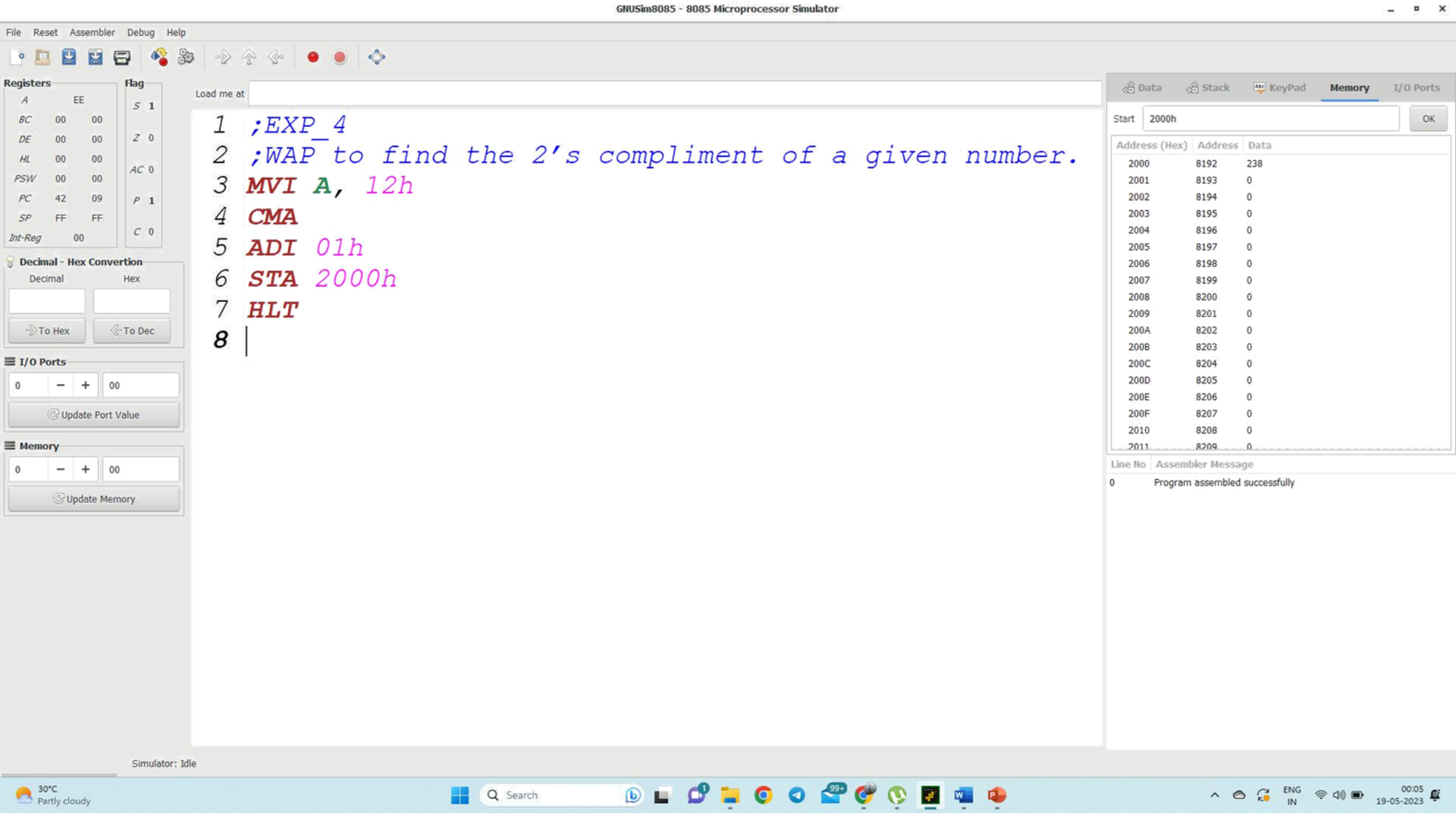
1. **First add contents of memory location 2000 and 2001 using "ADD" instruction and storing at 2000H.**
2. **The carry generated is recovered using JNC command and is stored at memory location 2001H.**

Source program: -

```
MVI A, 12h      // Move any no to the accumulator
CMA             // Compliment Accumulator
ADI 01h         // Add 01 to accumulator
STA 2000h // Display the result in memory location 2000h
HLT
```

RESULT

The 2's compliment of the given no is determined.


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
A EE
BC 00 00
DE 00 00
HL 00 00
PSW 00 00
PC 42 09
SP FF FF
Int-Reg 00
Flag
S 1
Z 0
AC 0
P 1
C 0
Decimal - Hex Convertion
Decimal
Hex
To Hex
To Dec
I/O Ports
0
00
Update Port Value
Memory
0
00
Update Memory
Load me at
1 ;EXP_4
2 ;WAP to find the 2's compliment of a given number.
3 MVI A, 12h
4 CMA
5 ADI 01h
6 STA 2000h
7 HLT
8
Data
Stack
KeyPad
Memory
I/O Ports
Start
2000h
OK
Address (Hex) Address Data
2000 8192 238
2001 8193 0
2002 8194 0
2003 8195 0
2004 8196 0
2005 8197 0
2006 8198 0
2007 8199 0
2008 8200 0
2009 8201 0
200A 8202 0
200B 8203 0
200C 8204 0
200D 8205 0
200E 8206 0
200F 8207 0
2010 8208 0
2011 8209 0
Line No Assembler Message
0 Program assembled successfully
Simulator: Idle
30°C
Partly cloudy
Search
ENG
IN
00:05
19-05-2023

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science Engineering

Practical No. 5

Aim:

WAP to transfer 5(51,52,53,54&55) numbers in to five successive memory location starting from 2000h onwards.

Software:

GNUSIM-8085

Theory

To transfer five consecutive number into 5 conservative memory location transfer one number into register and the starting memory location into memory pointer and then increase both one by one unless the whole data is transferred

achieved

Algorithm –

1. **First make starting memory address 2000h as memory pointer.**
2. **Store the 1st no in to Accumulator**
3. **Transfer the first no in to memory & increase memory pointer then transfer next no. do it 5 times. 001H.Run the loop for five times.**

Source program:-

```
LXI H, 2000h    // Move any address 2000h as memory pointer.
MVI A,051h      // Make register C as Counter
Loop; MOV M, A  // Transfer from Accumulator to memory
INX H           // Increment Memory
INR A  // Increment Accumulator
DCR C // Decrement Counter
JNZ: Loop
HLT
```

RESULT

The transfer of 5 consecutive data is done successfully.

GNUSim8085 - 8085 Microprocessor Simulator

File Reset Assembler Debug Help

Registers

A	33	
BC	00	00
DE	00	00
HL	21	00
PSW	00	00
PC	42	0D
SP	FF	FF
Int-Reg	00	

Flag

S 0 Z 1 AC 0 P 1 C 0

Decimal - Hex Convertion

Decimal: 51 Hex: 33 — To Hex / To Dec

I/O Ports

0 − + 00 — Update Port Value

Memory

2000 − + 33 — Update Memory

Load me at

```
1  ;EXP_5
2  ;WAP to transfer 5(51,52,53,54&55) numbers  in to
3  ;five successive memory location  starting from
4  ;2000h onwards.
5
6  LXI H, 2000h
7  MVI A,33H
8  Loop: MOV M,A
9  INX H
10 INR A
11 DCR C
12 JNZ Loop
13 HLT
14
```

Data | Stack | KeyPad | **Memory** | I/O Ports

Start OK

Address (Hex)	Address	Data
2000	8192	51
2001	8193	52
2002	8194	53
2003	8195	54
2004	8196	55
2005	8197	56
2006	8198	57
2007	8199	58
2008	8200	59
2009	8201	60
200A	8202	61
200B	8203	62
200C	8204	63
200D	8205	64
200E	8206	65
200F	8207	66
2010	8208	67
2011	8209	68

Line No	Assembler Message
0	Program assembled successfully

Simulator: Idle

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science Engineering

Practical No. 6

Aim:

WAP to add two 8 bit numbers and store the result at memory location 2000

Software:

GNUSIM-8085

Theory

The first data is brought to Accumulator A and the second one in any one of the other registers, say B. The addition is done using ADD. The result is then stored at 2000. The ADD instruction affects flags depending on result.

Algorithm –

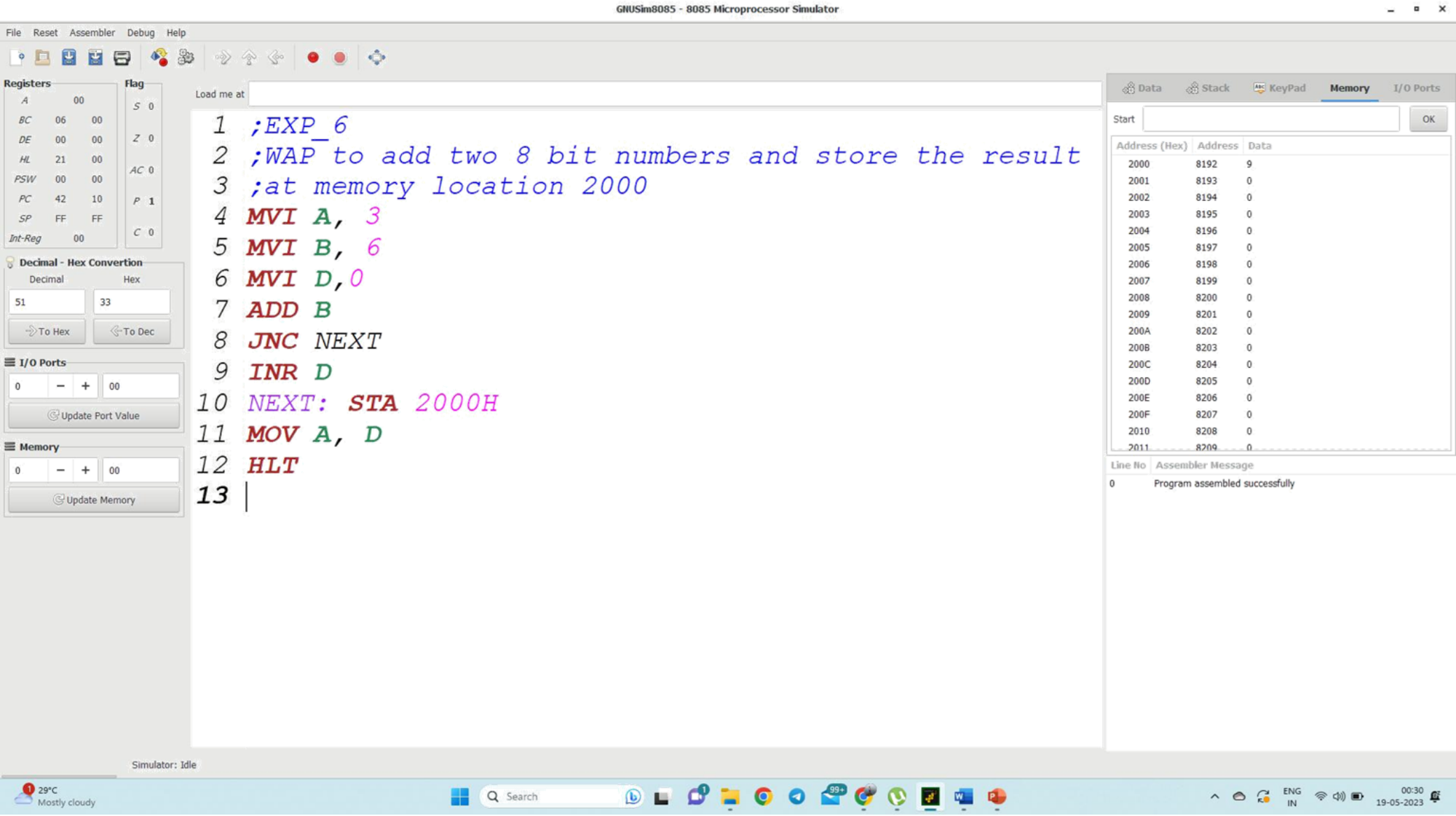
3. **First add contents of memory location 2000 and 2001 using "ADD" instruction and storing at 2000H.**
4. **The carry generated is recovered using JNC command and is stored at memory location 2001H.**

Source program:-

```
MVI A, 3 // Move immediate the value 3 in accumulator
MVI B, 6    // Move immediate the value 3 in register B
MVI D,0     // Initialize the value of D (used for carry)
ADD B       // Add the content of B with accumulator (A=A+B) and store the result in Accumulator
JNC NEXT    // (if carry=0, jump on NEXT)
INR D       // if carry=1 then incremnet the value of D by one
NEXT: STA 2000H //Store the value of Accumulator at memory location
2000 MOV A, D // Move the value of register D in the Accumulator (A)
HLT         //Terminate program execution
```

RESULT

Thus, the Addition of two 8-bit numbers is stored in the resultant memory


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
A 00
BC 06 00
DE 00 00
HL 21 00
PSW 00 00
PC 42 10
SP FF FF
Int-Reg 00
Flag
S 0
Z 0
AC 0
P 1
C 0
Decimal - Hex Convertion
Decimal
Hex
51
33
To Hex
To Dec
I/O Ports
0
00
Update Port Value
Memory
0
00
Update Memory
Load me at
1 ;EXP_6
2 ;WAP to add two 8 bit numbers and store the result
3 ;at memory location 2000
4 MVI A, 3
5 MVI B, 6
6 MVI D,0
7 ADD B
8 JNC NEXT
9 INR D
10 NEXT: STA 2000H
11 MOV A, D
12 HLT
13
Data
Stack
KeyPad
Memory
I/O Ports
Start
OK
Address (Hex) Address Data
2000 8192 9
2001 8193 0
2002 8194 0
2003 8195 0
2004 8196 0
2005 8197 0
2006 8198 0
2007 8199 0
2008 8200 0
2009 8201 0
200A 8202 0
200B 8203 0
200C 8204 0
200D 8205 0
200E 8206 0
200F 8207 0
2010 8208 0
2011 8209 0
Line No Assembler Message
0 Program assembled successfully
Simulator: Idle
29°C
Mostly cloudy
Search
ENG
IN
00:30
19-05-2023

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science Engineering

Practical No. 7

WAP to subtract two 8-bit numbers and store the result at memory location 2000

Software:

GNUSIM-8085

Theory

The first data is brought to Accumulator A and the second one in any one of the other registers, say B. The subtraction is done using SUB. The result is then stored at 2000.

Algorithm –

1. **First add contents of memory location 2000 and 2001 using "ADD" instruction and storing at 2000H.**
2. **The carry generated is recovered using JNC command and is stored at memory location 2001H.**

Source program: -

```
MVI A, 8        //Move immediate the value 8 in accumulator
MVI B, 4        //Move immediate the value 4 in register B
SUB B //Subtract the content of B with accumulator [redacted] and store the result in
Accumulator STA 2000H //Store the content of Accumulator in memory location 2000
HLT //Terminate program execution
```

RESULT

Thus, the Subtraction of two 8-bit numbers is stored in the resultant memory

GNUSim8085 - 8085 Microprocessor Simulator

File Reset Assembler Debug Help

Registers

A		04
BC	04	00
DE	00	00
HL	21	00
PSW	00	00
PC	42	09
SP	FF	FF
Int-Reg		00

Flag

S 0
Z 0
AC 0
P 0
C 0

Decimal - Hex Convertion

Decimal: 51 — To Hex
Hex: 33 — To Dec

I/O Ports

0 − + 00
Update Port Value

Memory

0 − + 00
Update Memory

Load me at

```
1  ;EXP_7
2  ;WAP to subtract two 8 bit numbers and store the
3  ; result at memory location 2000
4  MVI A, 8
5  MVI B, 4
6  SUB B
7  STA 2000H
8  HLT
9
```

Data | Stack | KeyPad | **Memory** | I/O Ports

Start 2000h OK

Address (Hex)	Address	Data
2000	8192	4
2001	8193	0
2002	8194	0
2003	8195	0
2004	8196	0
2005	8197	0
2006	8198	0
2007	8199	0
2008	8200	0
2009	8201	0
200A	8202	0
200B	8203	0
200C	8204	0
200D	8205	0
200E	8206	0
200F	8207	0
2010	8208	0
2011	8209	0

Line No	Assembler Message
0	Program assembled successfully

Simulator: Idle

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science & Engineering

Practical No. 8

Aim:

WAP to multiply two 8-bit numbers stored at memory location 2000 and 2001

Software:

GNUSIM-8085

Algorithm:

Store one of the data in a register (say C register). Move the second data to accumulator. Move the accumulator content to another register (say B register). Set the data in the C register as a counter. Set the accumulator value equal to zero. Add the data in B register to the content of accumulator. Decrement the value in C register. Repeat the addition until the value in the counter register C is zero. The final value in the accumulator will be the product of the two values.

Source program: -

STORE DATA AT MEMORY LOCATION

2000: 4

2001:5

LDA 2000H // Load Accumulator Directly from Memory

2000 MOV B, A // Copy the content of A (accumulator) in

B LDA 2001H // Load Accumulator Directly from Memory

2001 MOV C, A // Copy the content of A (accumulator)

in C (counter) MVI A, 0 // Initialize Accumulator=0

XYZ: ADD B // Loop Start: Add the content of Accumulator with B and store result in A

DCR C // Decrement the value of Register C by

one JNZ XYZ // If C is not equal to zero then

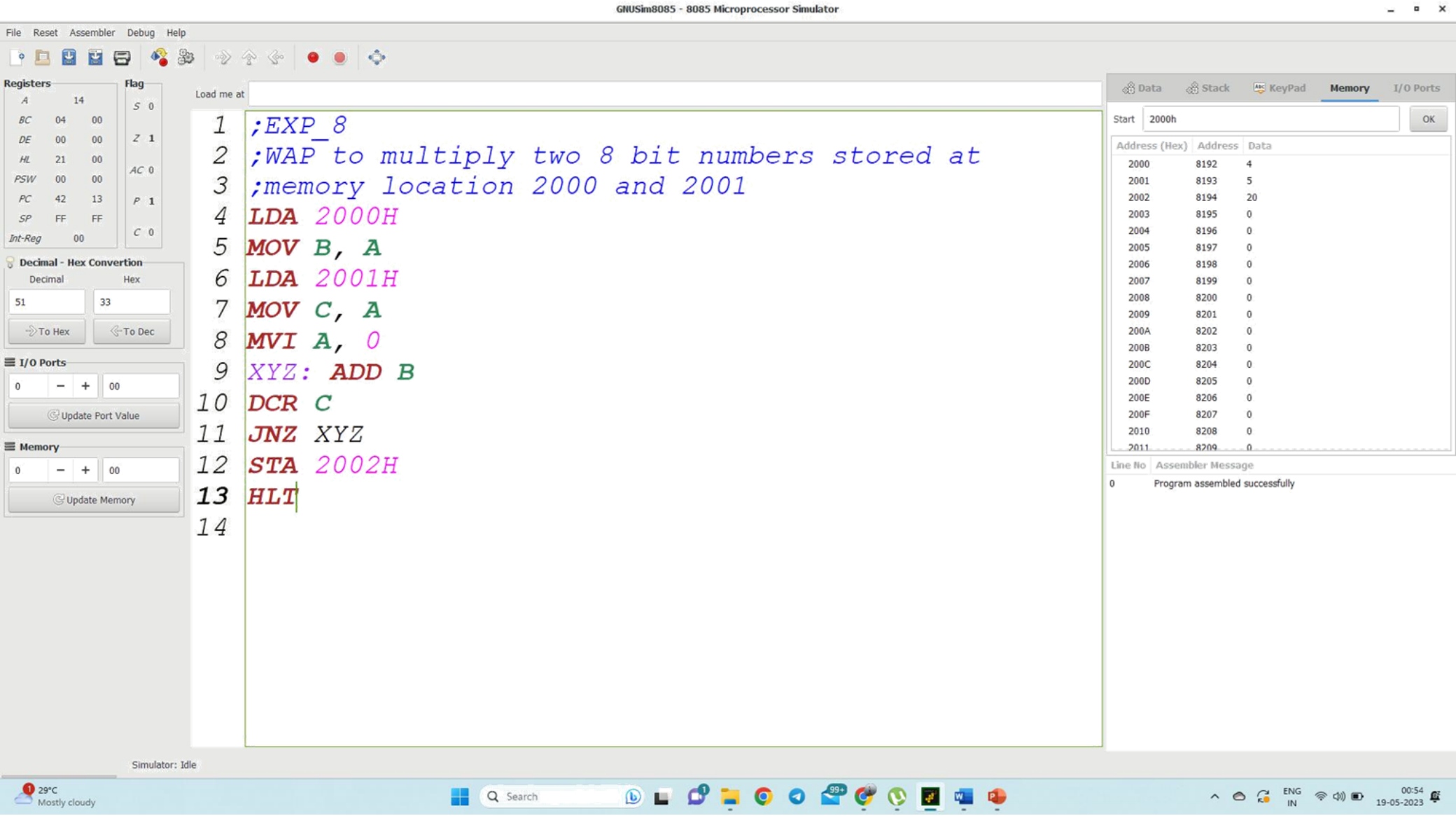
go on loop xyz

STA 2002H // Store the content of Accumulator (Result) in memory

location 2000 HLT // Terminate program execution

RESULT

Thus, the Multiplication of two 8-bit numbers is stored in the resultant memory


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
A 14
BC 04 00
DE 00 00
HL 21 00
PSW 00 00
PC 42 13
SP FF FF
Int-Reg 00
Flag
S 0
Z 1
AC 0
P 1
C 0
Decimal - Hex Convertion
Decimal
Hex
51
33
To Hex
To Dec
I/O Ports
0
00
Update Port Value
Memory
0
00
Update Memory
Load me at
1 ;EXP_8
2 ;WAP to multiply two 8 bit numbers stored at
3 ;memory location 2000 and 2001
4 LDA 2000H
5 MOV B, A
6 LDA 2001H
7 MOV C, A
8 MVI A, 0
9 XYZ: ADD B
10 DCR C
11 JNZ XYZ
12 STA 2002H
13 HLT
14
Data Stack KeyPad Memory I/O Ports
Start 2000h
OK
Address (Hex) Address Data
2000 8192 4
2001 8193 5
2002 8194 20
2003 8195 0
2004 8196 0
2005 8197 0
2006 8198 0
2007 8199 0
2008 8200 0
2009 8201 0
200A 8202 0
200B 8203 0
200C 8204 0
200D 8205 0
200E 8206 0
200F 8207 0
2010 8208 0
2011 8209 0
Line No Assembler Message
0 Program assembled successfully
Simulator: Idle
29°C
Mostly cloudy
Search
ENG
IN
00:54
19-05-2023

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science & Engineering

Practical No. 9

Aim:

WAP to add two 16-bit numbers. Store the result at memory address starting from 2000

Software:

GNUSIM-8085

Algorithm:

1. **We are adding two 16 bits number using DAD**
2. **Store the result in memory location 2000**

Source program: -

Store data at Memory location

4000: -6

4001: -2

4002: -5

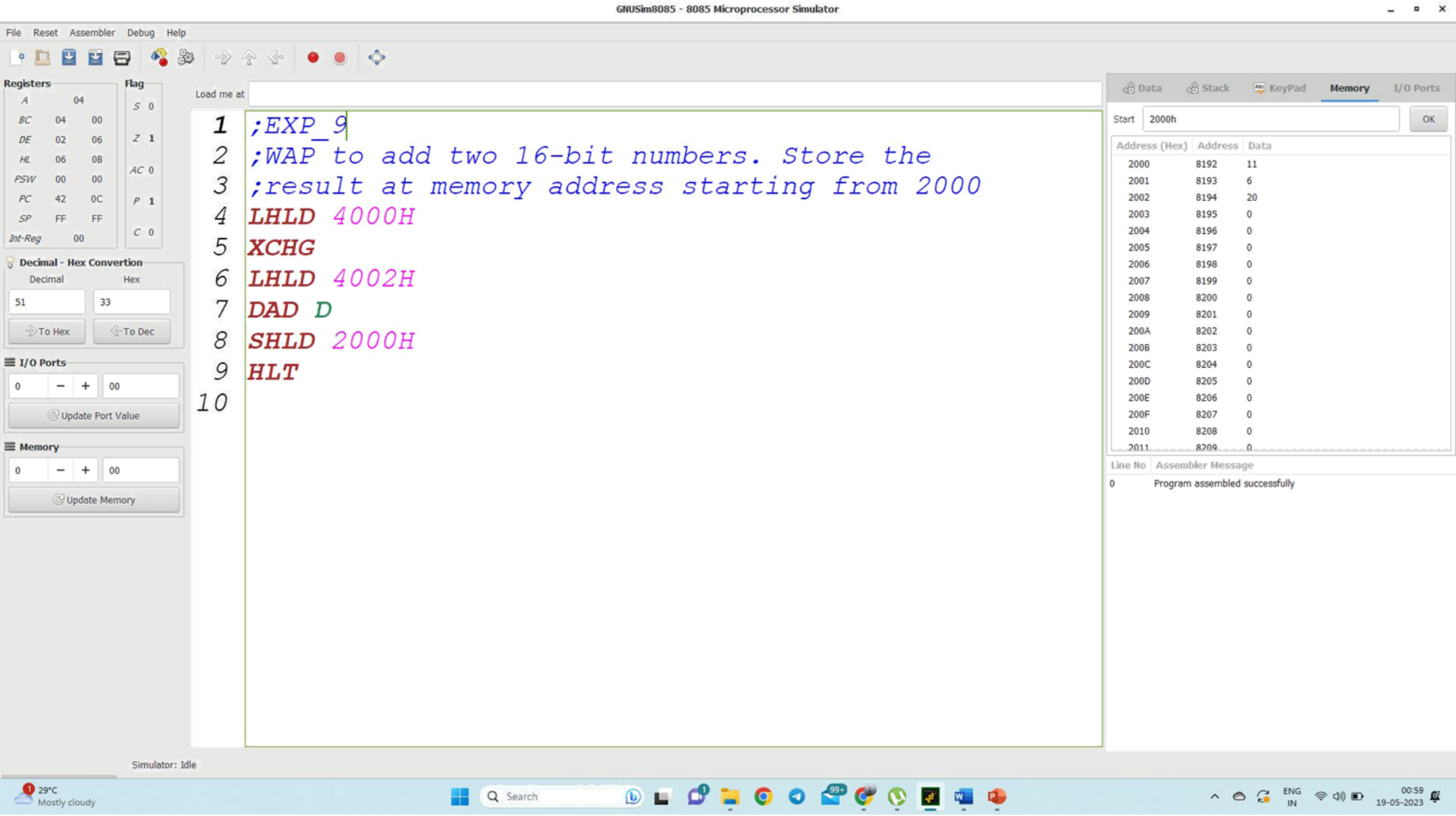
4003: -4

```
LHLD 4000H  //Load H & L Registers Directly from Memory 4000 L: -4000 and
XCHG         H: -4001 //Exchange H & L with D & E
LHLD 4002   //Load H & L Registers Directly from Memory 4002 L: -4002 and H: -4003
DAD D       //Double Register Add; Add Content of Register Pair H & L with register pair D
            &E and Store Result Register Pair H & L

SHLD 2000 //Store I6-bit result of Register Pair H & L in memory locations 2000
and 2001. HLT //Terminate program execution
```

Result: -

Thus, we have added two 16-bit numbers


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
A 04
BC 04 00
DE 02 06
HL 06 0B
PSW 00 00
PC 42 0C
SP FF FF
Int-Reg 00
Flag
S 0
Z 1
AC 0
P 1
C 0
Decimal - Hex Convertion
Decimal
Hex
51
33
To Hex
To Dec
I/O Ports
0
00
Update Port Value
Memory
0
00
Update Memory
Load me at
1 ;EXP_9
2 ;WAP to add two 16-bit numbers. Store the
3 ;result at memory address starting from 2000
4 LHLD 4000H
5 XCHG
6 LHLD 4002H
7 DAD D
8 SHLD 2000H
9 HLT
10
Data
Stack
KeyPad
Memory
I/O Ports
Start
2000h
OK
Address (Hex) Address Data
2000 8192 11
2001 8193 6
2002 8194 20
2003 8195 0
2004 8196 0
2005 8197 0
2006 8198 0
2007 8199 0
2008 8200 0
2009 8201 0
200A 8202 0
200B 8203 0
200C 8204 0
200D 8205 0
200E 8206 0
200F 8207 0
2010 8208 0
2011 8209 0
Line No Assembler Message
0 Program assembled successfully
Simulator: Idle
29°C
Mostly cloudy
Search
ENG
IN
00:59
19-05-2023

Gyan Ganga Institute of Technology and Sciences, Jabalpur

Computer Science & Engineering

Practical No. 10

Aim:

Assume that 3 bytes of data are stored at consecutive memory addresses of the data memory starting at 2000. Write a program which loads register C with (2000), i.e., with data contained at memory address 2000, D with (2001), E with (2002) and A with (2001).

Software:

GNUSIM-8085

Source program: -

```
Store data at Memory location
2000: -4
2001: -3
2002: -5
LDA 2000H   //Load Accumulator Directly from Memory 2000
MOV C, A    //Copy the content of A (accumulator) in C
LDA 2002H   //Load Accumulator Directly from Memory 2002
MOV E, A    //Copy the content of A (accumulator) in E //Load
LDA 2001H   Accumulator Directly from Memory 2001
MOV D, A    //Copy the content of A (accumulator) in D HLT //
Terminate program execution
```

Result: -

Thus, data is stored in register C, D, & E.

GNUSim8085 - 8085 Microprocessor Simulator

File Reset Assembler Debug Help

Registers

A		03
BC	00	04
DE	03	05
HL	00	00
PSW	00	00
PC	42	0D
SP	FF	FF
Int-Reg		00

Flag

S 0
Z 0
AC 0
P 0
C 0

Decimal - Hex Convertion

Decimal: 51 — To Hex
Hex: 33 — To Dec

I/O Ports

0 − + 00
Update Port Value

Memory

0 − + 00
Update Memory

Load me at

```
1  ;EXP_10
2  LDA 2000H
3  MOV C, A
4  LDA 2002H
5  MOV E, A
6  LDA 2001H
7  MOV D, A
8  HLT
9
```

Data | Stack | KeyPad | **Memory** | I/O Ports

Start 2000h OK

Address (Hex)	Address	Data
2000	8192	4
2001	8193	3
2002	8194	5
2003	8195	0
2004	8196	0
2005	8197	0
2006	8198	0
2007	8199	0
2008	8200	0
2009	8201	0
200A	8202	0
200B	8203	0
200C	8204	0
200D	8205	0
200E	8206	0
200F	8207	0
2010	8208	0
2011	8209	0

Line No	Assembler Message
0	Program assembled successfully

Simulator: Idle

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science & Engineering

Practical No. 11

Aim:

Sixteen bytes of data are specified at consecutive data-memory locations starting at 2000. Write a program which increments the value of all sixteen bytes by 01.

Software:

GNUSIM-8085

Source program: -

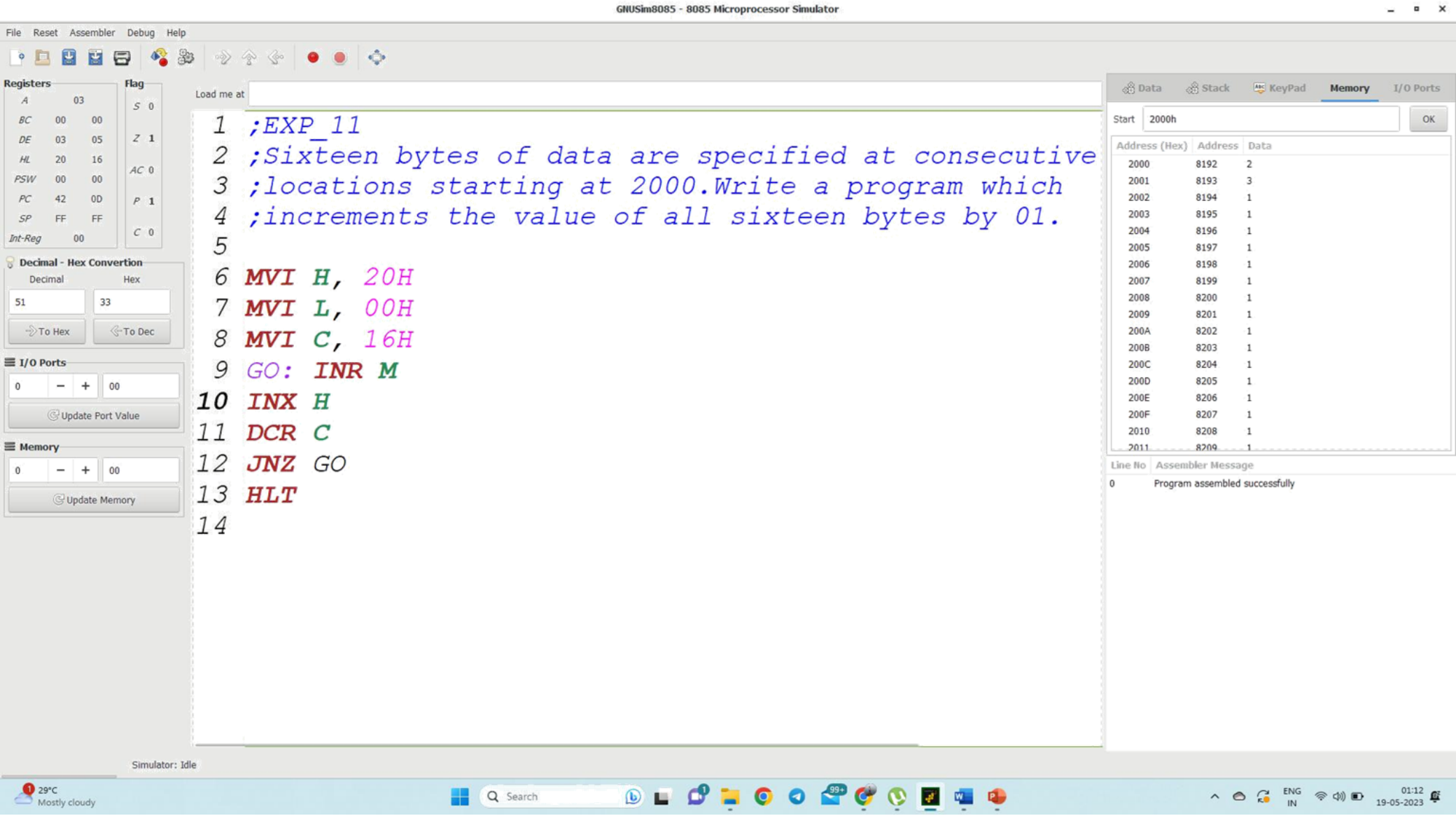
2000: -1
2001: -2
2002: -3

```
MVI H, 20H      //Move Immediate data 20 in Register H
MVI L, 00H //Move Immediate data 00 in Register L (Now HL pair is holding Memory Address 2000H)
MVI C, 16       //Move Immediate data 16 in Register C (C is counter)
//HL =2000
GO: INR M //Increment the value of M by 1 (M hold the value of Memory Location that HL pair Hold)

INX H 21 //Increment the value of register pair by one
DCR C //Decrement the value of Register C by one
JNZ GO          //If value of Register C is not equal to zero then go to loop xyz
HLT     // Terminate program execution
```

Result: -

Thus. We have incremented the value of all sixteen bytes by 01


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
A 03
BC 00 00
DE 03 05
HL 20 16
PSW 00 00
PC 42 0D
SP FF FF
Int-Reg 00
Flag
S 0
Z 1
AC 0
P 1
C 0
Decimal - Hex Convertion
Decimal
Hex
51
33
To Hex
To Dec
I/O Ports
0
00
Update Port Value
Memory
0
00
Update Memory
Load me at
1 ;EXP_11
2 ;Sixteen bytes of data are specified at consecutive
3 ;locations starting at 2000.Write a program which
4 ;increments the value of all sixteen bytes by 01.
5
6 MVI H, 20H
7 MVI L, 00H
8 MVI C, 16H
9 GO: INR M
10 INX H
11 DCR C
12 JNZ GO
13 HLT
14
Data
Stack
KeyPad
Memory
I/O Ports
Start
2000h
OK
Address (Hex) Address Data
2000 8192 2
2001 8193 3
2002 8194 1
2003 8195 1
2004 8196 1
2005 8197 1
2006 8198 1
2007 8199 1
2008 8200 1
2009 8201 1
200A 8202 1
200B 8203 1
200C 8204 1
200D 8205 1
200E 8206 1
200F 8207 1
2010 8208 1
2011 8209 1
Line No Assembler Message
0 Program assembled successfully
Simulator: Idle
29°C
Mostly cloudy
Search
ENG
IN
01:12
19-05-2023

Gyan Ganga Institute of Technology and Sciences, Jabalpur
Computer Science & Engineering

Practical No. 12

Aim:

WAP to add 10 bytes stored at memory location starting from 3000. Store the result at memory location 3000H

Software:

GNUSIM-8085

Algorithm:

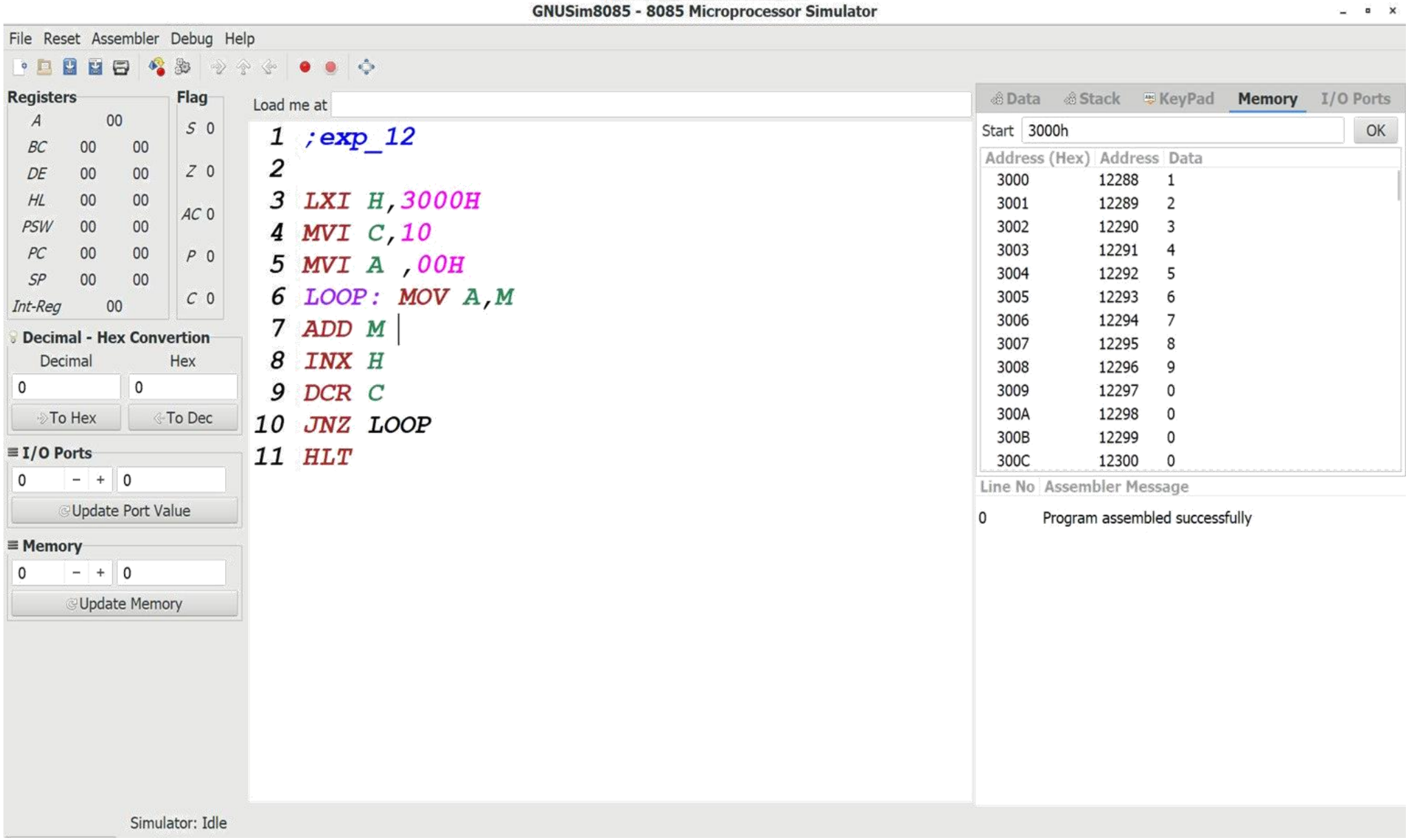
Move first data to accumulator. Initialize counter register. Add the next data with data in the accumulator. If there is a carry increments carry register. Decrement the county register. If it is zero store the result. Else fetch the next data and add with value in the accumulator. Repeat until carry register is zero.

Source program: -

```
MVI H,20H      //Move Immediate data 20 in Register H
MVIL,20H //Move Immediate data 00 in Register L (Now HL pair is holding
Memory Address 2000H)
MVI C,10       //Move Immediate data 16 in Register C (C is counter)
GO: ADD M //ADD the content of accumulator with M (M hold the value
of Memory Location that HL pair Hold)
INX H //Increment the value of register pair by one
DCR C //Decrement the value of Register C by one
JNZ GO //If value of Register C is not equal to zero then go
to loop xyz HLT // Terminate program execution
```

Result: -

Thus, we have added 10 bytes number and store the result at memory location 3000H.


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
A 00
BC 00 00
DE 00 00
HL 00 00
PSW 00 00
PC 00 00
SP 00 00
Int-Reg 00
Flag
S 0
Z 0
AC 0
P 0
C 0
Decimal - Hex Convertion
Decimal
Hex
0
0
To Hex
To Dec
I/O Ports
0 − + 0
Update Port Value
Memory
0 − + 0
Update Memory
Load me at
1 ;exp_12
2
3 LXI H,3000H
4 MVI C,10
5 MVI A ,00H
6 LOOP: MOV A,M
7 ADD M
8 INX H
9 DCR C
10 JNZ LOOP
11 HLT
Data Stack KeyPad Memory I/O Ports
Start 3000h OK
Address (Hex) Address Data
3000 12288 1
3001 12289 2
3002 12290 3
3003 12291 4
3004 12292 5
3005 12293 6
3006 12294 7
3007 12295 8
3008 12296 9
3009 12297 0
300A 12298 0
300B 12299 0
300C 12300 0
Line No Assembler Message
0 Program assembled successfully
Simulator: Idle