

# **COMPUTER ORGANISATION** **AND** **ARCHITECTURE LAB**

**ETCS - 260**



**Faculty: Mr. Varun Goel**

**Submitted by: Khushi**  
**Roll No: 03114813120**  
**Branch: ITE**  
**Semester: 4th**  
**Group: ITE-2**

## INDEX

S.No.	Experiment Name	Date of Submission	Teacher's sign/ Remarks
1.	To draw and explain (i) Block diagram and pin diagram of 8085 (ii) Instruction set of 8085	21/02/22	
2.	Write programs to perform: - (A) addition of two 8-bit binary numbers with carry. (B) addition of two 8-bit binary numbers without carry.	28/02/22	
3.	Write programs to perform: - (A) Subtraction of two 8-bit binary numbers with borrows. (B) Subtraction of two 8-bit binary numbers without borrows.	7/03/22	
4.	Finding one's complement of a number.	14/03/22	
5.	Finding Two's complement of a number.	21/03/22	
6.	Multiplication of two 8-bit binary numbers.	28/03/22	
7.	Find the smallest and largest numbers from the given series in 8085.	18/04/22	
8.	Calculate the sum of a series of numbers.	25/04/22	
9.	Find the factorial of a number.	2/05/22	

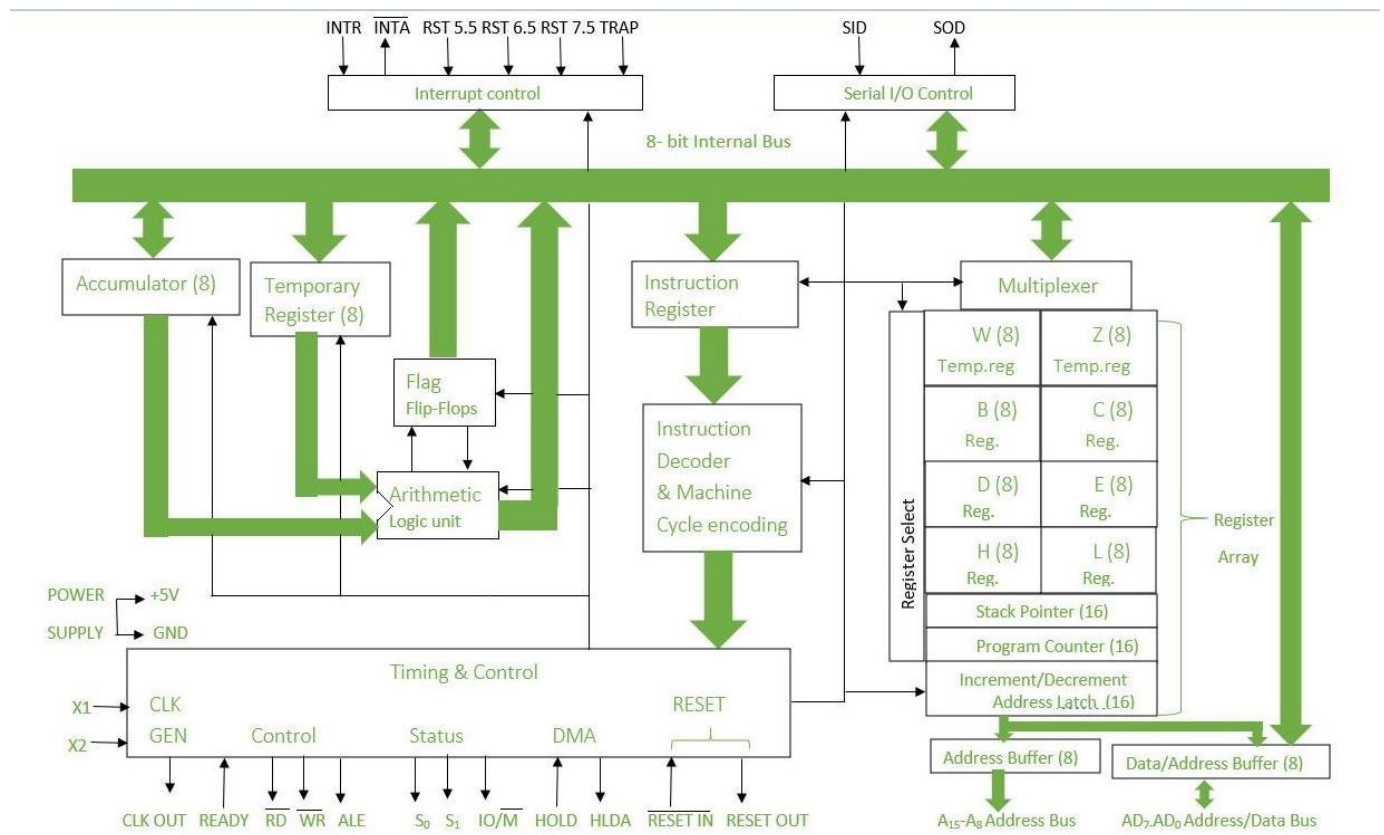
<b>10.</b>	Reverse an 8-bit number in 8085.	9/05/22	
<b>11.</b>	Sort an array in ascending and descending order in 8085.	23/05/22	
<b>12.</b>	Division of two 8 bit numbers in 8085	30/05/22	

# EXPERIMENT - 1

**AIM:** To draw and explain

(i) Block diagram and pin diagram of 8085

**Block diagram of 8085:**



## Arithmetic and Logic Unit (ALU):

It is used to perform mathematical operations like addition, multiplication, subtraction, division, decrement, increment, etc. Different operations are carried out in ALU: **Logical operations**, **Bit-Shifting Operations**, and **Arithmetic Operations**.

**Flag Register:** It is an 8-bit register that stores either 0 or 1 depending upon which value is stored in the accumulator. Flag Register contains 8-bit out of which 5-bits are important and the rest of 3-bits are Don't Care conditions. The flag register is a dynamic register because after each operation to check whether the result is zero, positive or negative whether there is any overflow occurred or not, or for comparison of two 8-bit numbers carry flag is checked. So for numerous operations to check the contents of the accumulator and from that contents, if we want to check the behavior of given result then we can use Flag register to verify and check.

Different fields of a flag register:

1. Carry Flag
2. Parity Flag
3. Auxiliary Carry Flag
4. Zero Flag
5. Sign Flag

**Accumulator:** Accumulator is used to performing I/O, arithmetic, and logical operations. It is connected to ALU and the internal data bus.

**General Purpose Register:** There are six general-purpose registers. These registers can hold 8-bit values. These 8-bit registers are B, C, D, E, H, L. These registers work as 16-bit registers when they work in pairs like B-C, D-E, and H-L. Here registers W and Z are reserved registers.

**Program Counter:** Program Counter holds the address value of the memory to the next instruction that is to be executed. It is a 16-bit register.

**Stack Pointer:** It works like a stack. In stack, the content of the register is stored that is later used in the program. It is a 16-bit special register. The stack pointer is part of memory but it is part of Stack operations, unlike random memory access. Stack pointer works in a continuous and contiguous part of the memory, whereas Program Counter works in random memory locations.

**Temporary Register:** It is an 8-bit register that holds data values during arithmetic and logical operations.

### **Instruction register and decoder:**

It is an 8-bit register that holds the instruction code that is being decoded. The instruction is fetched from the memory.

### **Timing and control unit:**

The timing and control unit comes under the CPU section, and it controls the flow of data from the CPU to other devices. It is also used to control the operations performed by the microprocessor and the devices connected to it. There are certain timing and control signals like Control signals, DMA Signals, RESET signals, and Status signals.

### **Interrupt control:**

Whenever a microprocessor is executing the main program and if suddenly an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program. There are 5 interrupt signals in 8085 microprocessors: INTR, TRAP, RST 7.5, RST 6.5, and RST 5.5.

**Priorities of Interrupts:** TRAP > RST 7.5 > RST 6.5 > RST 5.5 > INTR

### **Address bus and data bus:**

The data bus is bidirectional and carries the data which is to be stored. The address bus is unidirectional and carries the location where data is to be stored.

### **Serial Input/output control:**

It controls the serial data communication by using Serial input data and Serial output data.

### **The flow of an Instruction Cycle in 8085 Architecture:**

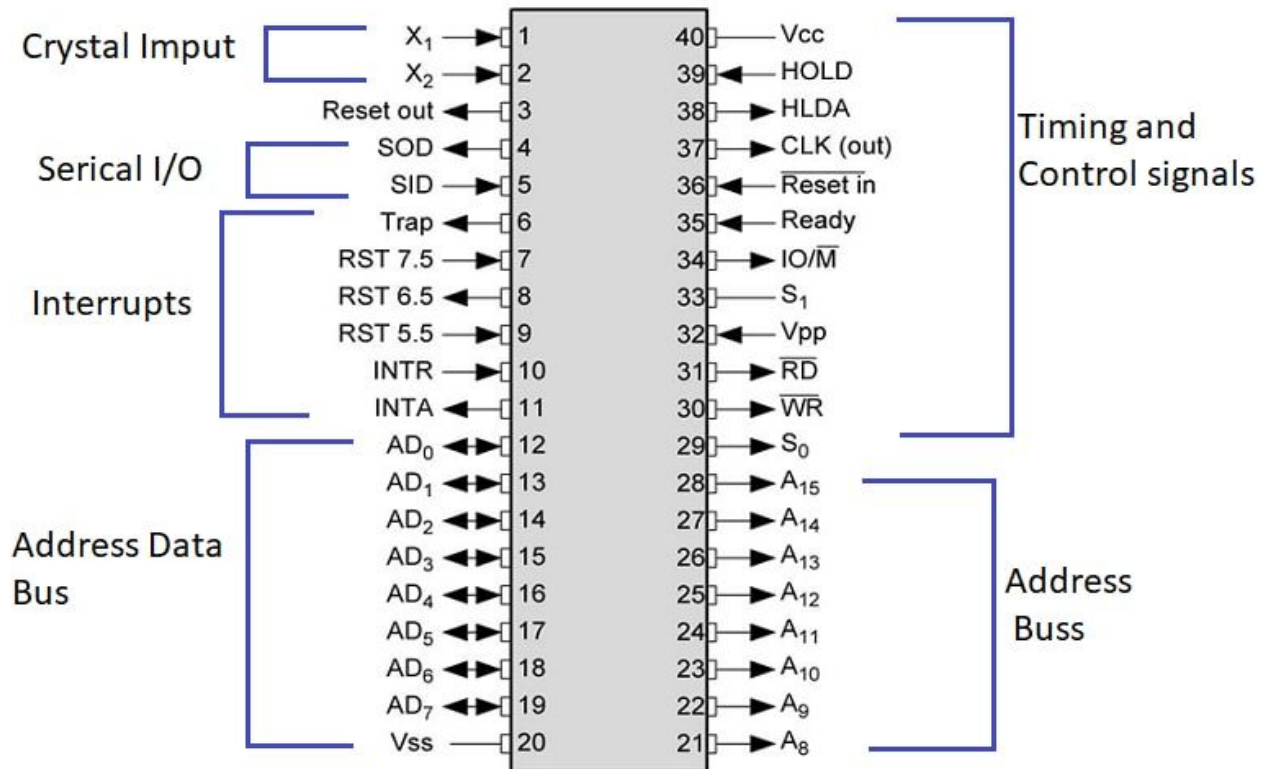
1. Execution starts with Program Counter. It starts program execution with the next address field. it fetches an instruction from the memory location pointed by Program Counter.
2. For address fetching from the memory, multiplexed address/data bus acts as an address bus and after fetching instruction this address bus will now acts as a data bus and extract data from the specified memory location and send this data on an

8-bit internal bus For multiplexed address/data bus Address Latch Enable(ALE) Pin is used. If **ALE = 1** (**Multiplexed bus is Address Bus otherwise it acts as Data Bus**).

3. After data fetching data will go into the Instruction Register it will store data fetched from memory and now data is ready for decoding so for this Instruction decoder register is used.
4. After that timing and control signal circuit comes into the picture. *It sends control signals all over the microprocessor to tell the microprocessor whether the given instruction is for READ/WRITE and whether it is for MEMORY/I-O Device activity.*
5. Hence according to timing and control signal pins, logical and arithmetic operations are performed and according to that data fetching from the different registers is done by a microprocessor, and mathematical operation is carried out by ALU. And according to operations Flag register changes dynamically.
6. With the help of Serial I/O data pin(SID or SOD Pins) we can send or receive input/output to external devices .in this way execution cycle is carried out.
7. ***While execution is going on if there is any interrupt detected then it will stop the execution of the current process and Invoke Interrupt Service Routine (ISR)*** Function. Which will stop the current execution and do execution of the current occurred interrupted after that normal execution will be performed.

## Pin Diagram of 8085:

### 8085 Pin Diagram



#### 1. Address Bus and Data Bus:

An address bus is a group of sixteen lines i.e A0-A15. The address bus is unidirectional, i.e., bits flow in one direction from the microprocessor unit to the peripheral devices and uses the high order address bus.

#### 2. Control and Status Signals:

- **ALE** – It is an Address Latch Enable signal. It goes high during first T state of a machine cycle and enables the lower 8-bits of the address, if its value is 1 otherwise data bus is activated.
- **IO/M'** – It is a status signal which determines whether the address is for input-output or memory. When it is high(1) the address on the address bus is for input-output devices. When it is low(0) the address on the address bus is for the memory.



- **SO, S1** – These are status signals. They distinguish the various types of operations such as halt, reading, instruction fetching or writing.
- **RD'** – It is a signal to control READ operation. When it is low the selected memory or input-output device is read.
- **WR'** – It is a signal to control WRITE operation. When it goes low the data on the data bus is written into the selected memory or I/O location.
- **READY** – It senses whether a peripheral is ready to transfer data or not. If READY is high(1) the peripheral is ready. If it is low(0) the microprocessor waits till it goes high. It is useful for interfacing low-speed devices.

### **3. Power Supply and Clock Frequency:**

- **Vss** – Ground Reference
- **Vcc** – +5v power supply
- **X1, X2** – A crystal is connected at these two pins. The frequency is internally divided by two, therefore, to operate a system at 3MHZ the crystal should have a frequency of 6MHZ.
- **CLK (OUT)** – This signal can be used as the system clock for other devices.

### **4. Interrupts and Peripheral Initiated Signals:**

The 8085 has five interrupt signals that can be used to interrupt program execution.

- (i) INTR
- (ii) RST 7.5
- (iii) RST 6.5
- (iv) RST 5.5
- (v) TRAP

The microprocessor acknowledges Interrupt Request by INTA signal. In addition to Interrupts, there are three externally initiated signals: RESET, HOLD, and READY. To respond to the HOLD requests, it has one signal called HLDA.

- **INTR** – It is an interrupt request signal.
- **INTA'** – It is an interrupt acknowledgment sent by the microprocessor after INTR is received.

## 5. Reset Signals:

- **RESET IN'** – When the signal on this pin is low(0), the program counter is set to zero, the buses are tri-stated and the microprocessor unit is reset.
- **RESET OUT** – This signal indicates that the MPU is being reset. The signal can be used to reset other devices.

## 6. DMA Signals:

- **HOLD** – It indicates that another device is requesting the use of the address and data bus. Having received the HOLD request the microprocessor relinquishes the use of the buses as soon as the current machine cycle is completed. Internal processing may continue. After the removal of the HOLD signal, the processor regains the bus.
- **HLDA** – It is a signal which indicates that the hold request has been received after the removal of a HOLD request, the HLDA goes low.

# EXPERIMENT - 2

**AIM:** Write programs to perform

i) addition of two 8-bit binary numbers with carry.

The screenshot shows the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window displays the assembly code for a program that adds two 8-bit numbers with carry. The registers and flags are shown on the left, and the memory and I/O ports are shown on the right.

**Registers:**

Register	Value
A	01
BC	01 00
DE	00 00
HL	00 00
PSW	00 00
PC	42 17
SP	FF FF
Int-Reg	00

**Flags:**

Flag	Value
S	0
Z	0
AC	0
P	0
C	0

**Assembly Code:**

```
1 ;Addition of two 8-bit number with carry
2
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11 MVI A,2000H
12 MVI B,2001H
13 MVI C,00H
14 ADD B
15 JNC Loop
16 INR C
17 Loop: STA 2005H
18 MOV B,A
19 STA 2003H
20
21 hlt
22
```

**Memory:**

Address (Hex)	Address	Data
2000	8192	192
2001	8193	244
2002	8194	0
2003	8195	1
2004	8196	0
2005	8197	1
2006	8198	0
2007	8199	0
2008	8200	0
2009	8201	0

**Assembler Message:**

Line No	Assembler Message
0	Program assembled successfully

ii) addition of two 8-bit binary numbers without carry.

The screenshot shows the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window displays the assembly code for a program that adds two 8-bit numbers without carry. The registers and flags are shown on the left, and the memory and I/O ports are shown on the right.

**Registers:**

Register	Value
A	96
BC	32 00
DE	00 00
HL	00 96
PSW	00 00
PC	42 10
SP	FF FF
Int-Reg	00

**Flags:**

Flag	Value
S	1
Z	0
AC	0
P	1
C	0

**Assembly Code:**

```
1 ;<Program title>
2
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11
12 LDA 2050
13 MOV B, A
14 LDA 2051
15 ADD B
16 STA 2052
17 HLT
18
19
20
21
```

**Memory:**

Address (Hex)	Address	Data
0802	2050	50
0803	2051	100
0804	2052	150
0805	2053	0
0806	2054	0
0807	2055	0
0808	2056	0
0809	2057	0
080A	2058	0
080B	2059	0

**Assembler Message:**

Line No	Assembler Message
0	Program assembled successfully

# EXPERIMENT - 3

**AIM:** Write programs to perform

i) subtraction of two 8-bit binary numbers without borrow.

The screenshot shows the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window displays the assembly code for a program titled "<Program title>". The code is as follows:

```
1 ;<Program title>
2
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11
12 LXI H, 4000H
13 MOV A, M
14 INX H
15 SUB M
16 INX H
17 MOV M, A
18 HLT
```

The left panel shows the Registers window with the following values:

Register	Value
A	32
BC	00 00
DE	00 00
HL	40 02
PSW	00 00
PC	42 0D
SP	FF FF
Int-Reg	00

The right panel shows the Memory window with the following data:

Address (Hex)	Address	Data
4000	16384	50
4001	16385	0
4002	16386	50
4003	16387	0
4004	16388	0
4005	16389	0
4006	16390	0
4007	16391	0
4008	16392	0
4009	16393	0

The bottom panel shows the I/O Ports and Memory windows, both with values 0 and 00.

ii) subtraction of two 8-bit binary numbers with borrow.

The screenshot shows the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window displays the assembly code for a program titled "Subtraction of two 8-bit number with borrow". The code is as follows:

```
1 ;Subtraction of two 8-bit number with borrow
2
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11 MVI A, 2000H
12 MVI B, 2001H
13 MVI C, 00H
14 SUB B
15 JNC Loop
16 INR C
17 Loop: STA 2005H
18 MOV A, B
19 STA 2003H
20
21
22 hlt
```

The left panel shows the Registers window with the following values:

Register	Value
A	01
BC	01 01
DE	00 00
HL	00 00
PSW	00 00
PC	42 17
SP	FF FF
Int-Reg	00

The right panel shows the Memory window with the following data:

Address (Hex)	Address	Data
2000	8192	131
2001	8193	255
2002	8194	0
2003	8195	1
2004	8196	0
2005	8197	255
2006	8198	0
2007	8199	0
2008	8200	0
2009	8201	0

The bottom panel shows the I/O Ports and Memory windows, both with values 0 and 00.

# EXPERIMENT - 4

**AIM:** Finding one's complement of a number.

The screenshot displays the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window is titled "GNUSim8085 - 8085 Microprocessor Simulator" and includes a menu bar (File, Reset, Assembler, Debug, Help) and a toolbar with various icons.

**Registers and Flags:** A table on the left shows the status of registers and flags.

Registers	Flag
A	S 0
BC 00 00	Z 0
DE 00 00	AC 0
HL 00 00	P 0
PSW 00 00	C 0
PC 42 0C	
SP FF FF	
Int-Reg 00	

**Assembly Code:** The central area displays the following code:

```
1  
2 ;<Program title>  
3  
4 jmp start  
5  
6 ;data  
7  
8  
9 ;code  
10 start: nop  
11  
12 LDA 4400H  
13 CMA  
14 STA 4401H  
15 HLT
```

**Memory:** A table on the right shows the memory contents.

Address (Hex)	Address	Data
4400	17408	50
4401	17409	205
4402	17410	0
4403	17411	0
4404	17412	0
4405	17413	0
4406	17414	0
4407	17415	0
4408	17416	0
4409	17417	0

**I/O Ports:** A section at the bottom left shows the I/O Ports with a value of 0 and a button to "Update Port Value".

**Assembler Message:** A message box at the bottom right indicates "Program assembled successfully".

# EXPERIMENT - 5

**AIM:** Finding two's complement of a number

The screenshot displays the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window is titled "GNUSim8085 - 8085 Microprocessor Simulator" and includes a menu bar (File, Reset, Assembler, Debug, Help) and a toolbar with various icons.

**Registers:** The left panel shows the status of 8085 registers. The Accumulator (A) contains 00, and the Carry Flag (C) is set to 0. Other registers like BC, DE, HL, PSW, PC, and SP are also shown with their values.

**Assembly Code:** The central pane displays the following assembly code:

```
1 ;<Program title>
2
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11
12 LDA 4500H
13 CMA
14 ADI 01H
15 STA 4501H
16 HLT
```

**Memory:** The right panel shows the memory dump. The start address is 4500h. The memory contains the following data:

Address (Hex)	Address	Data
4500	17664	50
4501	17665	206
4502	17666	0
4503	17667	0
4504	17668	0
4505	17669	0
4506	17670	0
4507	17671	0
4508	17672	0
4509	17673	0

**I/O Ports:** The bottom left panel shows the I/O Ports section with a value of 0 and a button to "Update Port Value".

**Assembler Message:** The bottom right panel shows the assembler message: "Program assembled successfully".

# EXPERIMENT - 6

**AIM:** Multiplication of two 8-bit binary numbers.

The screenshot displays the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window shows the assembly code for a program to perform multiplication of two 8-bit numbers. The code is as follows:

```
1 ;<Program to perform multiplication of two 8 bit numbers>
2
3
4 jmp start
5
6 ;data
7
8 ;code
9 start: nop
10
11 LXI H, 2050H
12 MOV B, M
13 INX H
14 MOV C, M
15 MVI A, 00H
16 TOP: ADD B
17 DCR C
18 JNZ TOP
19 INX H
20 MOV M, A
21
22 hlt
23
```

The left panel shows the Registers window with the following values:

Register	Value
A	1E
BC	05 00
DE	00 00
HL	20 52
PSW	00 00
PC	42 14
SP	FF FF
Int-Reg	00

The right panel shows the Memory window with the following data:

Address (Hex)	Address	Data
2050	8272	5
2051	8273	6
2052	8274	30
2053	8275	0
2054	8276	0
2055	8277	0
2056	8278	0
2057	8279	0
2058	8280	0
2059	8281	0
205A	8282	0
205B	8283	0
205C	8284	0
205D	8285	0

The bottom status bar indicates "Simulator: Idle".





# EXPERIMENT - 8

**AIM:** Write a program to find the sum of a series of n consecutive numbers.

The screenshot displays the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window shows assembly code being entered line by line. The code includes a program title, a jump to start, data declaration, code segment, and two loops for summing consecutive numbers. The registers section on the left shows the current state of the 8085 registers. The memory section on the right shows a dump of memory addresses and their corresponding data values. The status bar at the bottom indicates the simulator is idle.

**Registers:**

Register	Value
A	37
BC	37 02
DE	00 00
HL	80 02
PSW	00 00
PC	42 1F
SP	FF FF
Int-Reg	00

**Flag:**

Flag	Value
S	0
Z	1
AC	0
P	1
C	0

**Assembly Code:**

```
1 ;<Program title>
2
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11
12 LDA 201BH
13 MOV B, A
14 INR A
15 MOV C, A
16 MVI A, 00H
17 LOOP1: ADD B
18 DCR C
19 JNZ LOOP1
20 MVI C, 02H
21 MVI B, 00H
22 LOOP2: INR B
23 SUB C
24 JNZ LOOP2
25 MOV A, B
26 STA 201CH
27
28 hlt
29
```

**Memory Dump:**

Address (Hex)	Address	Data
201B	8219	10
201C	8220	55
201D	8221	0
201E	8222	0
201F	8223	0
2020	8224	0
2021	8225	0
2022	8226	0
2023	8227	0
2024	8228	0
2025	8229	0
2026	8230	0
2027	8231	0
2028	8232	0

**Assembler Message:**

```
Line No Assembler Message
0 Program assembled successfully
```

# EXPERIMENT - 9

**AIM:** Find the factorial of a number.

The screenshot displays the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window is divided into several sections:

- Registers:** A table showing the status of various registers. The PC (Program Counter) is at 42, and the SP (Stack Pointer) is at FF.
- Flag:** A section showing the status of various flags. The S (Sign) flag is 0, Z (Zero) is 1, AC (Auxiliary Carry) is 0, P (Parity) is 1, and C (Carry) is 0.
- Decimal - Hex Conversion:** A section for converting between decimal and hex values. The decimal value 3050 is entered, and the hex value BEA is shown.
- I/O Ports:** A section for interacting with I/O ports. The port value 0 is entered, and the 'Update Port Value' button is visible.
- Memory:** A section for managing memory. The memory address 5001 is entered, and the 'Update Memory' button is visible.
- Assembly Code:** A central area for writing and executing assembly code. The code is as follows:

```
1 ;<Program title>
2
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11
12 LXI H, 2000H
13 MOV B, M
14 MVI D, 01H
15 FACTORIAL: CALL MULTIPLY
16 DCR B
17 JNZ FACTORIAL
18 INX H
19 MOV M, D
20 HLT
21 MULTIPLY: MOV E, B
22 XRA A
23 ML: ADD D
24 DCR E
25 JNZ ML
26 MOV D, A
27 RET
28
29
30
31
```
- Memory Window:** A table showing the contents of memory. The address 2000 is selected, and the data is 5. The table shows the following data:

Address (Hex)	Address	Data
2000	8192	5
2001	8193	120
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
- Assembler Message:** A section showing the output of the assembler. The message "Program assembled successfully" is displayed.

# EXPERIMENT - 10

**AIM:** Reverse an 8-bit number in 8085.

The screenshot displays the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window is titled "GNUSim8085 - 8085 Microprocessor Simulator" and includes a menu bar (File, Reset, Assembler, Debug, Help) and a toolbar with various icons.

**Registers:** The left panel shows the state of the 8085 registers. The A register contains 80. The BC, DE, HL, PSW, PC, and SP registers are shown with their values. The Int-Reg register is 00. The Flag section shows the status of the S, Z, AC, P, and C flags.

**Decimal - Hex Conversion:** Below the registers, there is a section for converting between decimal and hex values. The decimal input is 0, and the hex output is 0. Buttons for "To Hex" and "To Dec" are available.

**I/O Ports:** The I/O Ports section shows the current port value as 0. Buttons for increment (+), decrement (-), and update are present.

**Memory:** The Memory section shows the current memory address as 0. Buttons for increment (+), decrement (-), and update are present.

**Assembly Code:** The central pane displays the assembly code for reversing an 8-bit number. The code is as follows:

```
1 Load me at [ ]
2 ;Reverse a 8-bit number
3
4 jmp start
5
6 ;data
7
8
9 ;code
10 start: nop
11
12 LDA 2050
13 RLC
14 RLC
15 RLC
16 RLC
17 STA 2055
18
19
20 hlt
```

**Memory Dump:** The right panel shows a memory dump starting at address 2050. The data is as follows:

Address (Hex)	Address	Data
0802	2050	8
0803	2051	0
0804	2052	0
0805	2053	0
0806	2054	0
0807	2055	128
0808	2056	0
0809	2057	0
080A	2058	0
080B	2059	0

**Assembler Message:** The bottom right pane shows the assembler message: "Program assembled successfully".

**Simulator Status:** The status bar at the bottom indicates "Simulator: Idle".

# EXPERIMENT - 11

**AIM:** i) Sort an array in ascending order in 8085.

The screenshot shows the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window displays the assembly code for sorting an array in ascending order. The code is as follows:

```

1  ;<Program title>
2
3
4  jmp start
5
6  ;data
7
8
9  ;code
10 start: nop
11
12
13
14      LXI H,5000
15      MOV C,M
16      DCR C
17  REPEAT: MOV D,C
18           LXI H,5001
19  LOOP:  MOV A,M
20           INX H
21           CMP M
22           JC SKIP
23           MOV B,M
24           MOV M,A
25           DCX H
26           MOV M,B
27           INX H
28  SKIP:  DCR D
29           JNZ LOOP
30           DCR C
31           JNZ REPEAT
32           HLT

```

The left panel shows the Registers window with the following values:

Register	Value	Flag
A	01	S 0
BC	01 00	Z 1
DE	00 00	AC 0
HL	13 8A	P 1
PSW	00 00	C 1
PC	42 21	
SP	FF FF	
Int-Reg	00	

The right panel shows the Memory window with the following data:

Address (Hex)	Address	Data
1388	5000	5
1389	5001	1
138A	5002	2
138B	5003	3
138C	5004	4
138D	5005	5
138E	5006	0
138F	5007	0
1390	5008	0
1391	5009	0
1392	5010	0
1393	5011	0

The bottom status bar indicates "Simulator: Idle".

ii) Sort an array in descending order in 8085.

The screenshot shows the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window displays the assembly code for sorting an array in descending order. The code is as follows:

```

1  ;<Program title>
2
3
4  jmp start
5
6  ;data
7
8
9  ;code
10 start: nop
11
12
13
14      LXI H,5000
15      MOV C,M
16      DCR C
17  REPEAT: MOV D,C
18           LXI H,5001
19  LOOP:  MOV A,M
20           INX H
21           CMP M
22           JNC SKIP
23           MOV B,M
24           MOV M,A
25           DCX H
26           MOV M,B
27           INX H
28  SKIP:  DCR D
29           JNZ LOOP
30           DCR C
31           JNZ REPEAT
32           HLT

```

The left panel shows the Registers window with the following values:

Register	Value	Flag
A	05	S 0
BC	05 00	Z 1
DE	00 00	AC 0
HL	13 8A	P 1
PSW	00 00	C 0
PC	42 21	
SP	FF FF	
Int-Reg	00	

The right panel shows the Memory window with the following data:

Address (Hex)	Address	Data
1388	5000	5
1389	5001	5
138A	5002	4
138B	5003	3
138C	5004	2
138D	5005	1
138E	5006	0
138F	5007	0
1390	5008	0
1391	5009	0
1392	5010	0
1393	5011	0

The bottom status bar indicates "Simulator: Idle".

# EXPERIMENT - 12

**AIM:** division of two 8-bit numbers in 8085.

The screenshot displays the GNUSim8085 - 8085 Microprocessor Simulator interface. The main window is divided into several sections:

- Registers:** A table showing the status of 8085 registers. The Accumulator (A) contains 01, BC is 05 01, DE is 00 00, HL is 13 89, PSW is 00 00, PC is 42 1D, SP is FF FF, and Int-Reg is 00. Flags S, Z, AC, P, and C are also shown.
- Decimal - Hex Conversion:** A section with input fields for decimal (5000) and hex (1388) values, and buttons to convert between them.
- I/O Ports:** A section with a value of 0 and buttons to update the port value.
- Memory:** A section with a value of 5001 and buttons to update memory.
- Assembly Code:** A central area showing the assembly program being executed. The code includes instructions like `LXI H, 5000`, `MOV B, M`, `MVI C, 00`, `INX H`, `MOV A, M`, `NEXT: CMP B`, `JC LOOP`, `SUB B`, `INR C`, `JMP NEXT`, `LOOP: STA 5002`, `MOV A, C`, `STA 5003`, and `HLT`.
- Memory Window:** A table on the right showing memory addresses (hex) and data. The address 1389 contains the value 5, and address 138A contains 0. The address 138B contains 1, and address 138C contains 0. The address 138D contains 0, and address 138E contains 0. The address 138F contains 0, and address 1390 contains 0. The address 1391 contains 0, and address 1392 contains 0. The address 1393 contains 0.
- Assembler Message:** A section at the bottom right showing the message "Program assembled successfully".

The status bar at the bottom indicates "Simulator: Idle".