CERTIFICATE

*This is to certify that Mr./Ms. **…..... Hemil…Chovatiya............*** *with enrolment no.* ***..........200303108003.................*** *has successfully completed **his**/her laboratory experiments in the **Computer Organization & Architecture Lab(203105254)** from the department of **........Information Technology(4ITA1)….….........*** *during the academic year **........2021-2022.........***

Date of Submission: .... Staff In charge: ....

Head of Department: ....

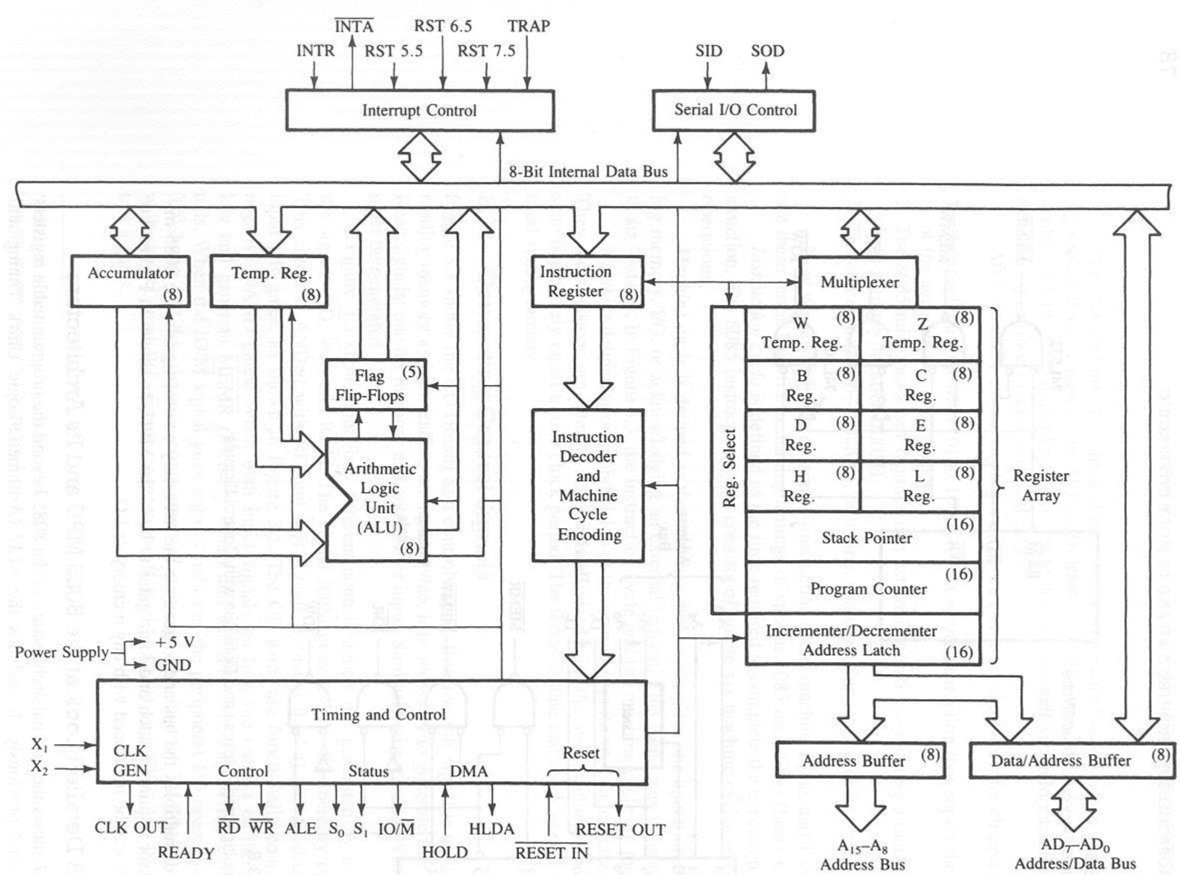
INDEX

Sr. No	Experiment Title	Page No		Date of Performance	Date of Assessment	Marks (out of 10)	Sign
		From	To				
1	Write the working of 8085 simulator GNUsim8085 and basic architecture of 8085 along with small introduction.						
2	Study the complete instruction set of 8085 and write the instructions in the instruction set of 8085along with examples.						
3	Write an assembly language code in GNUsim8085 to implement Addition of two 8bit Numbers.						
4	Write an assembly language code in GNUsim8085 to implement Addition of two 16 bitNumbers.						
5	Write an assembly language code in GNUsim8085 to implement Multiplication of two 8bitNumbers.						
6	Write an assembly language code in GNUsim8085 to implement Division of two 8bit Numbers.						
7	Write an assembly language code in GNUsim8085 to add two 8 bit numbers stored in memory andalso storing the carry.						
8	Write an assembly language code in GNUsim8085 to store numbers in reverse order in memorylocation.						

PRACTICAL 1

Aim: Write the working of 8085 simulator GNUsim8085 and basic architecture of 8085along with small introduction.

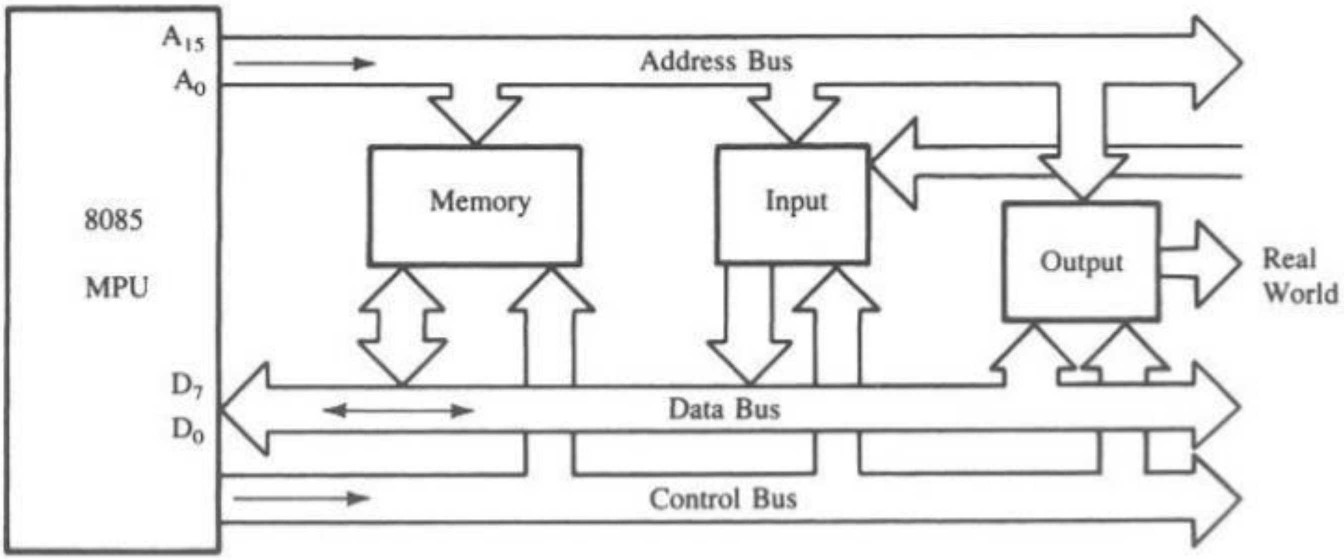
- **8085 Microprocessor Architecture**



- The architecture of the 8085 microprocessor mainly includes the timing & control unit, Arithmetic and logic unit, decoder, instruction register, interrupt control, a register array, serial input/output control.
- The most important part of the microprocessor is the central processing unit.

- **The 8085 Bus Structure**

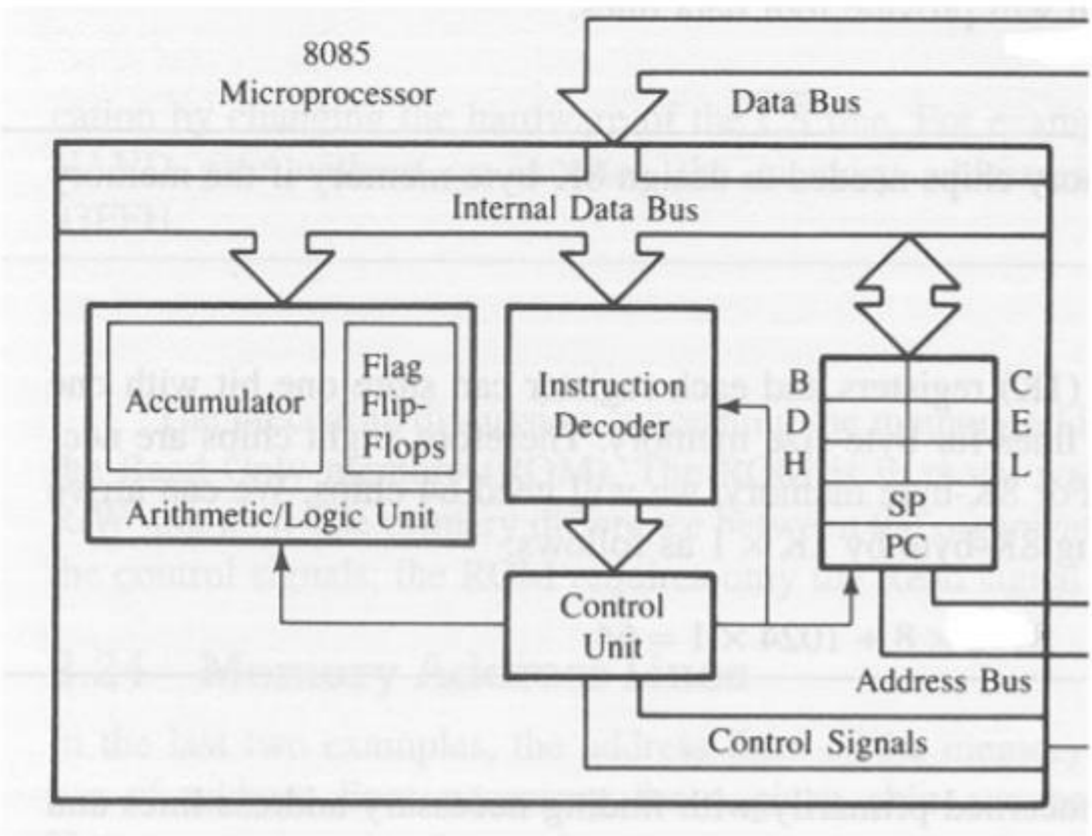
The 8-bit 8085 CPU (or MPU – Micro Processing Unit) communicates with the other units using a16-bit address bus, an 8-bit data bus and a control bus.



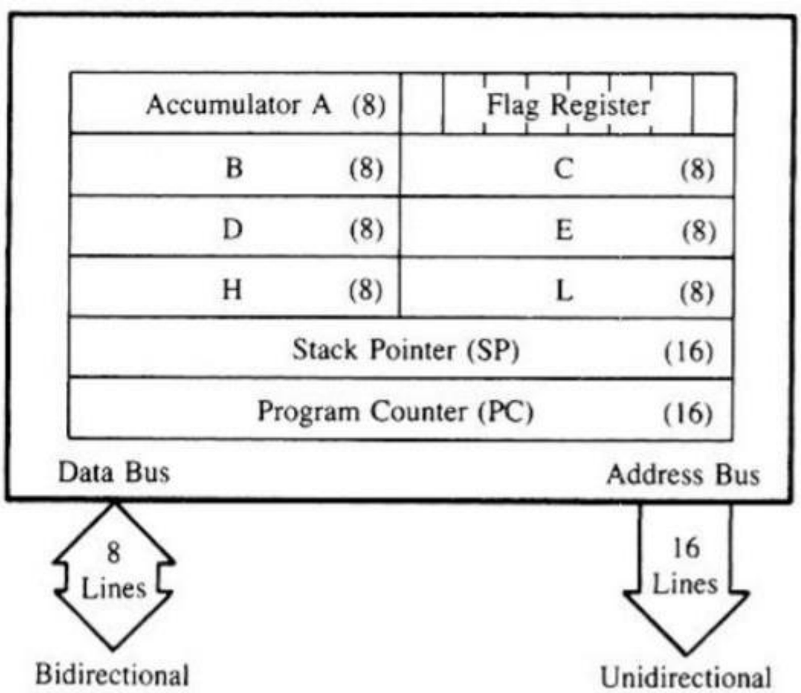
- **Address Bus**
  - Consists of 16 address lines: A0 – A15
  - Operates in unidirectional mode: The address bits are always sent from the MPU to peripheral devices, not reverse.
  - 16 address lines are capable of addressing a total of 216 = 65,536 (64k) memory locations.
  - Address locations: 0000 (hex) – FFFF (hex)

- **Data Bus**
  - Consists of 8 data lines: D0 – D7
  - Operates in bidirectional mode: The data bits are sent from the MPU to peripheral devices, as well as from the peripheral devices to the MPU.
  - Data range: 00 (hex) – FF (hex)
- **Control Bus**
  - Consists of various lines carrying the control signals such as read / write enable, flag bits.

- **The 8085: CPU Internal Structure**
  - The internal architecture of the 8085 CPU is capable of performing the following operations: Store 8-bit data (Registers, Accumulator).
  - Perform arithmetic and logic operations (ALU)
  - Test for conditions (IF / THEN)
  - Sequence the execution of instructions
  - Store temporary data in RAM during execution



- **The 8085: Registers**



  - Six general purpose 8-bit registers: B, C, D, E, H, L

- They can also be combined as register pairs to perform 16-bit operations: BC, DE, HL
- Registers are programmable (data load, move, etc.)

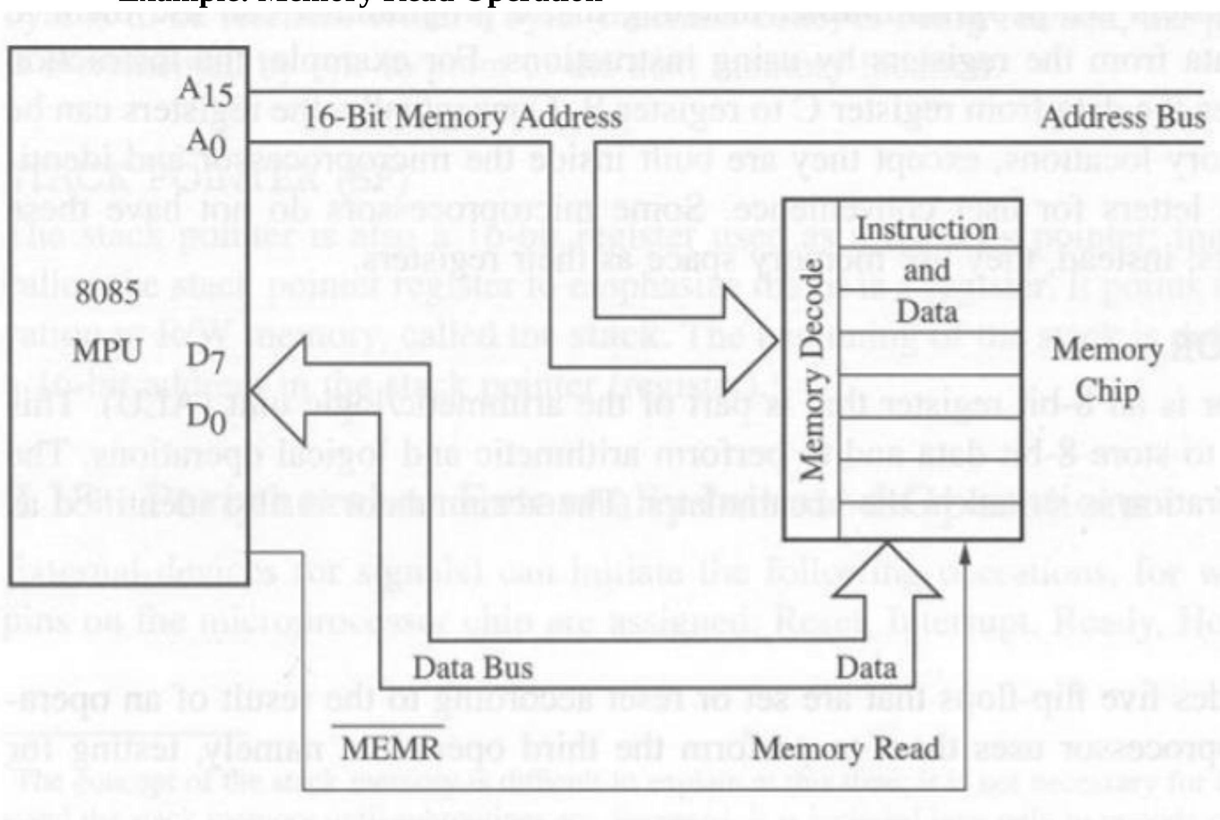
- **Accumulator**
  - Single 8-bit register that is part of the ALU !
  - Used for arithmetic / logic operations – the result is always stored in the accumulator.

- **Flag Bits**
  - Indicate the result of condition tests.
  - Carry, Zero, Sign, Parity, etc.
  - Conditional operations (IF / THEN) are executed based on the condition of these flag bits.

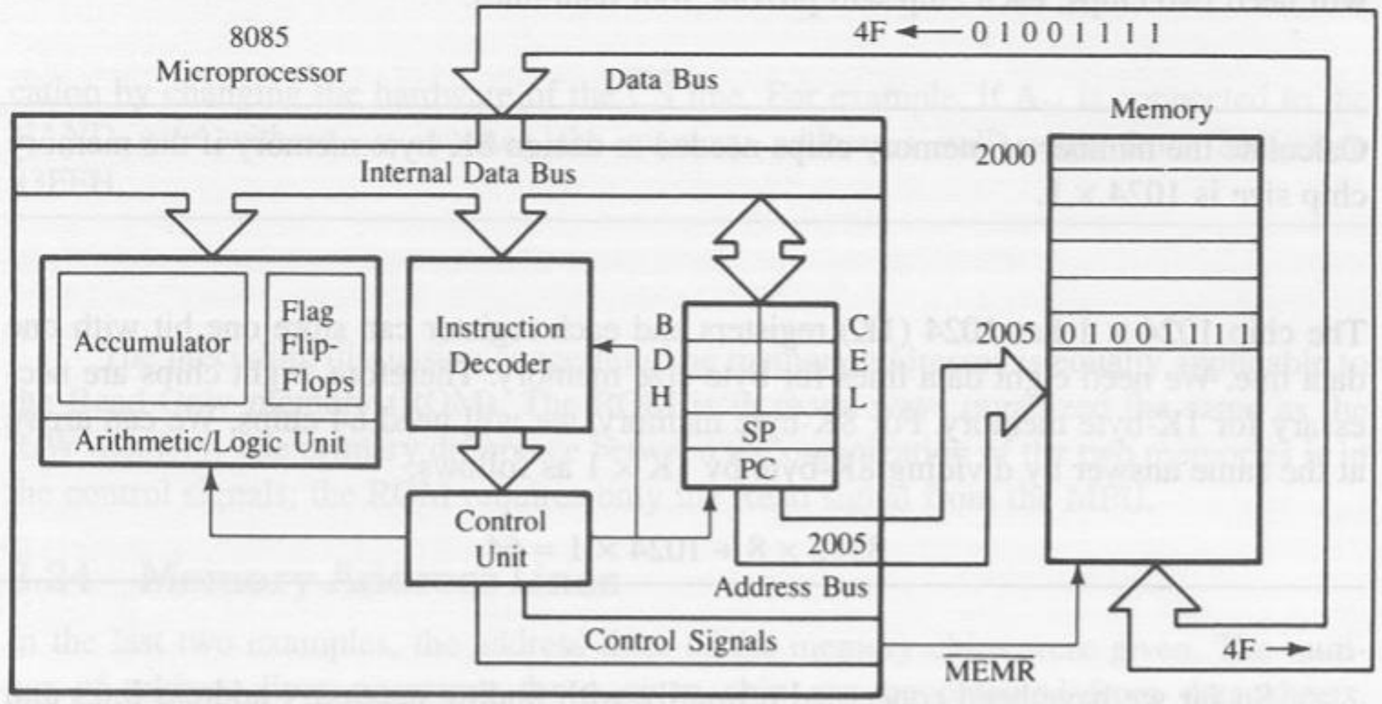
- **Program Counter (PC)**
  - Contains the memory address (16 bits) of the instruction that will be executed in the next step.

- **Stack Pointer (SP)**
  - Stack pointer is a special purpose 16-bit register in the Microprocessor, which holds the address of the top of the stack.

- **Example: Memory Read Operation**



- **Example: Instruction Fetch Operation**
  - All instructions (program steps) are stored in memory.
  - To run a program, the individual instructions must be read from the memory in sequence, and executed.
  - Program counter puts the 16-bit memory address of the instruction on the address bus
  - Control unit sends the Memory Read Enable signal to access the memory
  - The 8-bit instruction stored in memory is placed on the data bus and transferred to the instruction decoder
  - Instruction is decoded and executed



PRACTICAL 2

Aim: Study the complete instruction set of 8085 and write the instructions with the instruction set along with examples

Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called Instruction Set.
- 8085 has 246 instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value are called Op-Code or Instruction Byte.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

Data Transfer Instructions

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

Data Transfer Instructions

Opcode	Operand	Description
MOV	Rd, Rs Rd, MM, Rs	Copy from source to destination.

- This instruction copies the contents of the source register into the destination register. The contents of the source register are not altered.
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.
- Example: MOV B, C
- MOV B, M
- MOV M, C

Opcode	Operand	Description
MVI	Rd, DataM, Data	Move immediate 8-bit

- The 8-bit data is stored in the destination register or memory.
- If the operand is a memory location, its location is specified by the contents of the H-L registers. Example: MVI A, 57H
- MVI M, 57H

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

- This instruction loads 16-bit data in the register pair.
- Example: LXI H, 2034 H

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- The contents of a memory location, specified by a 16- bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
- Example: LDA 2034H

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.

- The contents of either the register pair or the memory location are not altered.
- Example: LDAX B

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of the next memory location into register H.
- Example: LHLD 2040 H

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of the accumulator are copied into the memory location specified by the operand.
- Example: STA 2500 H

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

- The contents of the accumulator are copied into the memory location specified by the contents of the register pair.
- Example: STAX B

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.
- Example: SHLD 2550 H

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.
- Example: XCHG

Arithmetic Instructions

These instructions perform the operations like:

- Addition
- Subtract
- Increment
- Decrement

Opcode	Operand	Description
ADD	RM	Add register or memory to accumulator

- The contents of the register or memory are added to the contents of the accumulator.
- The result is stored in the accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- Example: ADD B or ADD M

Opcode	Operand	Description
ADC	RM	Add register or memory to accumulator with carry

- The contents of register or memory and Carry Flag (CY) are added to the contents of the accumulator.
- The result is stored in the accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- Example: ADC B or ADC M

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

- The 8-bit data is added to the contents of the accumulator.
- The result is stored in the accumulator.
- All flags are modified to reflect the result of the addition.
- Example: ADI 45 H

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

- The 8-bit data and the Carry Flag (CY) are added to the contents of the accumulator.
- The result is stored in the accumulator.
- All flags are modified to reflect the result of the addition.
- Example: ACI 45 H

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The 16-bit contents of the register pair are added to the contents of the H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY is set.
- No other flags are changed.
- Example: DAD B

Opcode	Operand	Description
SUB	RM	Subtract register or memory from accumulator

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in the accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- Example: SUB B or SUB M

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

- The 8-bit data is subtracted from the contents of the
- accumulator.
- The result is stored in the accumulator.
- All flags are modified to reflect the result of subtraction.
- Example: SUI 45 H

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in the accumulator.
- All flags are modified to reflect the result of subtraction.
- Example: SBI 45 H

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- Example: INR B or INR M

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of the register pair are incremented by 1.
- The result is stored in the same place.
- Example: INX H

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- The contents of register or memory location are decremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- Example: DCR B or DCR M

Logical Instructions

- These instructions perform logical operations on data stored in registers, memory and status flags.

The logical operations are:

- AND
- OR
- XOR
- ROTATE
- COMPARE
- COMPLEMENT

Opcode	Operand	Description
CMP	RM	Compare register or memory with accumulator

- The contents of the operand (register or memory) are compared with the contents of the accumulator.
- Both contents are preserved .
- The result of the comparison is shown by setting the flags of the PSW as follows:
- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.
- Example: CMP B or CMP M

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- The 8-bit data is compared with the contents of the accumulator.
- The values being compared remain unchanged.
- The result of the comparison is shown by setting the flags of the PSW as follows:
- if (A) < data: carry flag is set
- if (A) = data: zero flag is set
- if (A) > data: carry and zero flags are reset
- Example: CPI 89H

Opcode	Operand	Description
ANA	RM	Logical AND register or memory with accumulator

- The contents of the accumulator are logically ANDed with the contents of register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY is reset and AC is set.
- Example: ANA B or ANA M.

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

- The contents of the accumulator are logically ANDed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY is reset, AC is set.
- Example: ANI 86H.

Opcode	Operand	Description
XRA	RM	Exclusive OR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.

- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- Example: XRA B or XRA M.

Opcode	Operand	Description
ORA	RM	Logical OR register or memory with accumulator

- The contents of the accumulator are logically ORed with the contents of the register or
- memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result, CY and AC are reset.
- Example: ORA B or ORA M.

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

- The contents of the accumulator are logically ORed with the 8-
- bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- Example: ORI 86H.

Opcode	Operand	Description
XRA	RM	Logical XOR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- Example: XRA B or XRA M.

Branching Instructions

- The branching instruction alter the normal sequential flow.
- These instructions alter either unconditionally or conditionally.

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- Example: JMP 2034 H.

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- Example: JZ 2034 H.

Jump Conditionally

Opcode	Operand	Description
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
- Example: CALL 2034 H.

Opcode	Operand	Description
Cx	16-bit address	Call unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.
- Example: CZ 2034 H.

Call Conditionally

Opcode	Operand	Description
CC	Call if Carry	CY = 1
CNC	Call if No Carry	CY = 0
CP	Call if Positive	S = 0
CM	Call if Minus	S = 1
CZ	Call if Zero	Z = 1
CNZ	Call if No Zero	Z = 0
CPE	Call if Parity Even	P =1
CPO	Call if Parity Odd	P = 0

Opcode	Operand	Description
RET	None	Return unconditionally

- The program sequence is transferred from the subroutine to the calling program.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- Example: RET.

Opcode	Operand	Description
Rx	None	Call conditionally

- The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- Example: RZ.

Return Conditionally

Opcode	Operand	Description
RC	Return if Carry	CY = 1
RNC	Return if No Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if No Zero	Z = 0
RPE	Return if Parity Even	P =1
RPO	Return if Parity Odd	P = 0

Opcode	Operand	Description
RST	0-7	Restart (Software Interrupts)

- The RST instruction jumps the control to one of eight memory locations depending upon the number.
- These are used as software instructions in a program to transfer program execution to one of the eight locations.
- Example: RST 3.

Restart Address Table

Instructions	Restart Address
RST 0	0000 H
RST 1	0008 H
RST 2	0010 H
RST 3	0018 H
RST 4	0020 H
RST 5	0028 H
RST 6	0030 H
RST 7	0038 H

Control Instructions

Opcode	Operand	Description
NOP	None	No operation

- No operation is performed.
- The instruction is fetched and decoded but no operation is executed.
- Example: NOP

Opcode	Operand	Description
HLT	None	Halt

- The CPU finishes executing the current instruction and halts any further execution.
- An interrupt or reset is necessary to exit from the halt state.
- Example: HLT

Opcode	Operand	Description
DI	None	Disable interrupt

- The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled.
- No flags are affected.
- Example: DI

Opcode	Operand	Description
EI	None	Enable interrupt

- The interrupt enable flip-flop is set and all interrupts are enabled.
- No flags are affected.
- This instruction is necessary to re-enable the interrupts (except TRAP).
- Example: EI

Opcode	Operand	Description
RIM	None	Read interrupt mask

- This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data inputbit.
- The instruction loads eight bits in the accumulator with the following interpretations.
- Example: RIM

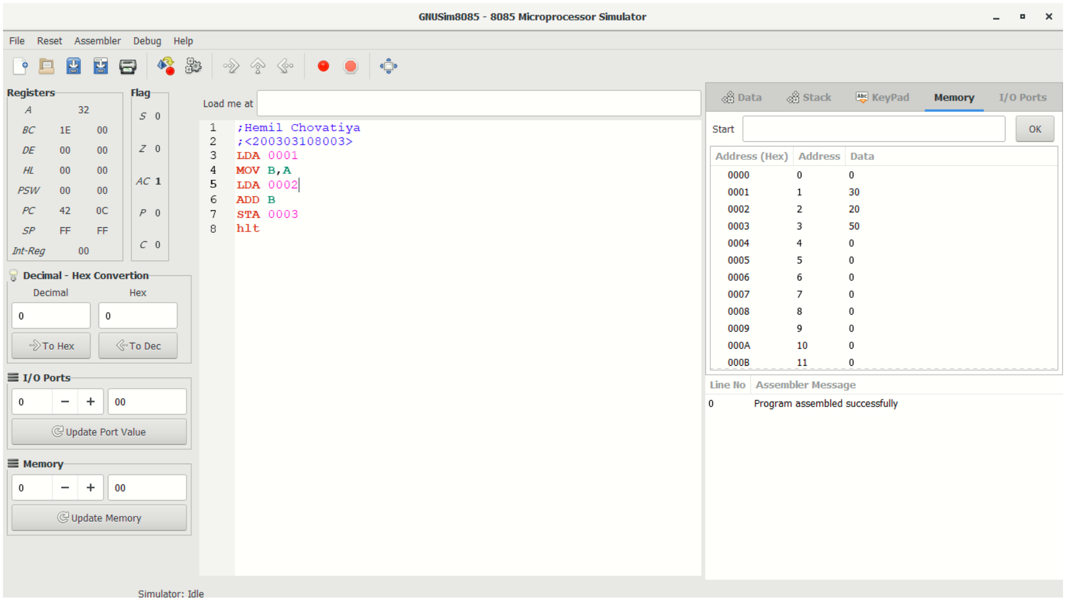
PRACTICAL 3

AIM: Write an assembly language code in GNUsim8085 to implement Addition of two 8 bit Numbers.

Theory:

Code	Meaning
LDA 0001	Load value of memory location 0001 in Accumulator A
MOV B,A	Move data from memory to accumulator
STA 0003	Store accumulator contents in memory
ADD B	Add data of memory with accumulator
HLT	Hold the program

Implementation:



Input:

0001 = 30

0002 = 20

Output:

0003 = 50

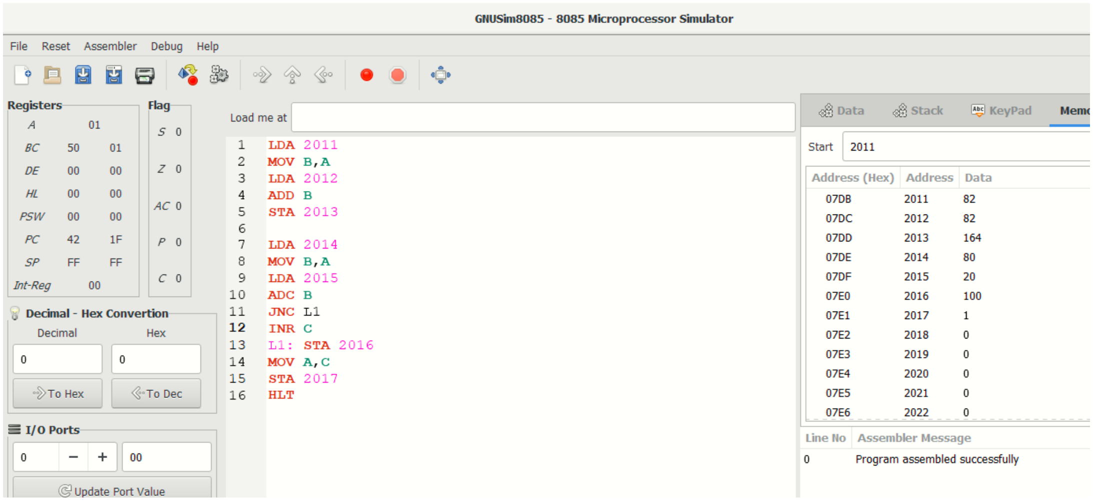
PRACTICAL 4

AIM: Write an assembly language code in GNUsim8085 to implement Addition of two 16-bit Numbers.

Theory:

Code	Meaning
LDA 2011	Load value of memory location 0001 in Accumulator A
MOV B, A	Move data from memory to accumulator
STA 0003	Store accumulator contents in memory
JNC L1	Jump if no carry CY = 0
INR C	Increment register or memory by 1
ADC B	Add register or memory to accumulator with carry
ADD B	Add data of memory with accumulator
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2011=82
2012=82
2014=80
2015=20

OUTPUT: 2013=164
2016=100
2017=2

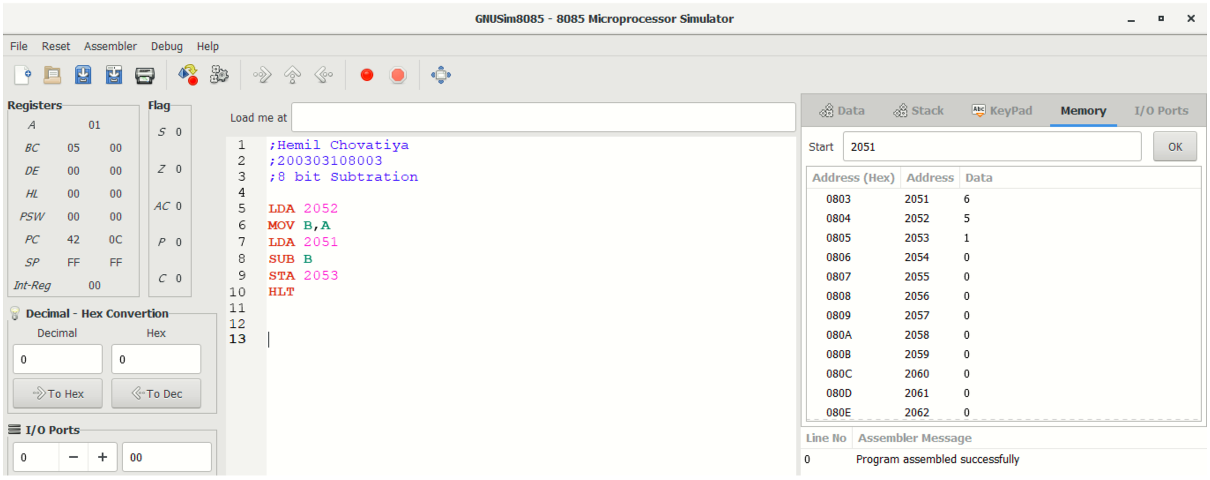
PRACTICAL 5

AIM: Write an assembly languagecode in GNUsim8085 to implement Subtraction of two 8bitNumbers.

THEORY

Code	Meaning
LDA 2052	Load value of memory location 2011 in Accumulator A
MOV B, A	Move data from memory to accumulator
STA 2053	Store accumulator contents in memory
SUB B	Subtract register or memoryfrom accumulator
HLT	Hold the program

Implementation:



Input:

2051 = 6
2052 = 5

Output:

2053 = 1

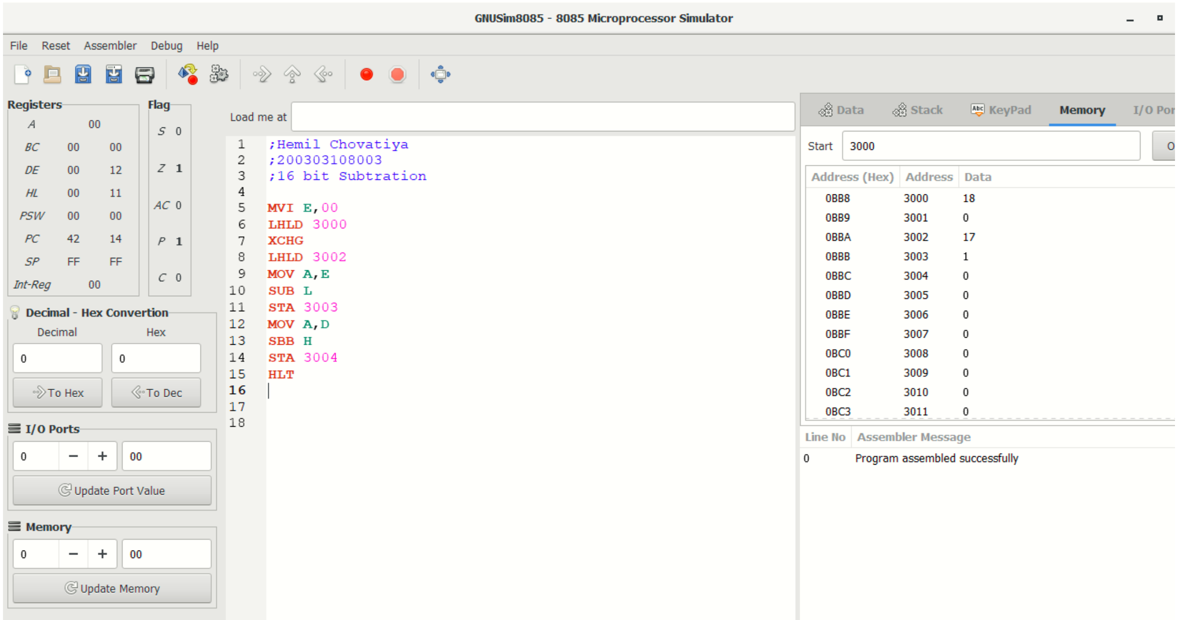
PRACTICAL 6

AIM: Write an assembly languagecode in GNUsim8085 to implement Subtraction of two 16bitNumbers.

THEORY

Code	Meaning
LDA 2052	Load value of memory location 2011 in Accumulator A
MOV B, A	Move data from memory to accumulator
STA 2053	Store accumulator contents in memory
SUB B	Subtract register or memoryfrom accumulator
HLT	Hold the program
MVI A,00	8-bit data is stored in the destination register or memory. (Move immediate 8-bit)
LHLD	Load H-L registers direct

Implementation:



Input:

3000 = 18

3002 = 17

Output:

3003 = 1

3004 = 0

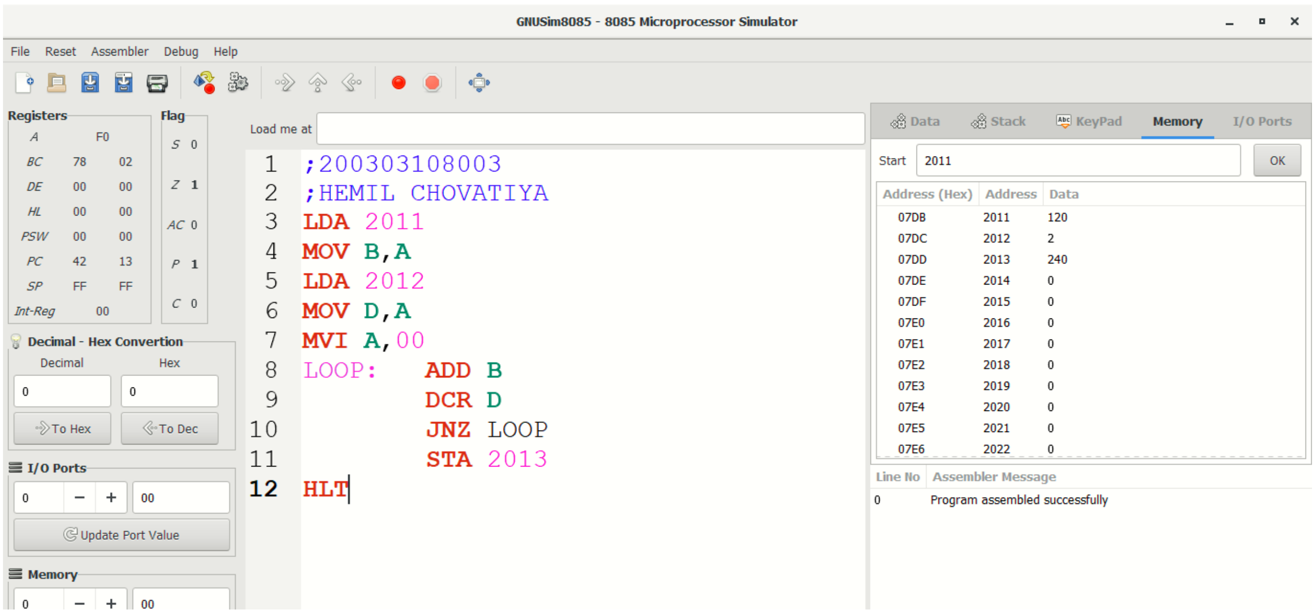
PRACTICAL 7

AIM: Write an assembly languagecode in GNUsim8085 to implement Multiplication of two 8bitNumbers.

THEORY

Code	Meaning
LDA 2011	Load value of memory location 2011 in Accumulator A
MOV B, A	Move data from memory to accumulator
STA 2013	Store accumulator contents in memory
MVI A,00	8-bit data is stored in the destination register or memory. (Move immediate 8-bit)
DCR D	Decrement register or memory by 1.
JNZ LOOP	Jump if No Zero (Z = 0) to LOOP
ADD B	Add data of memory with accumulator
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2011 = 120
2012 = 2

OUTPUT: 2013 = 240

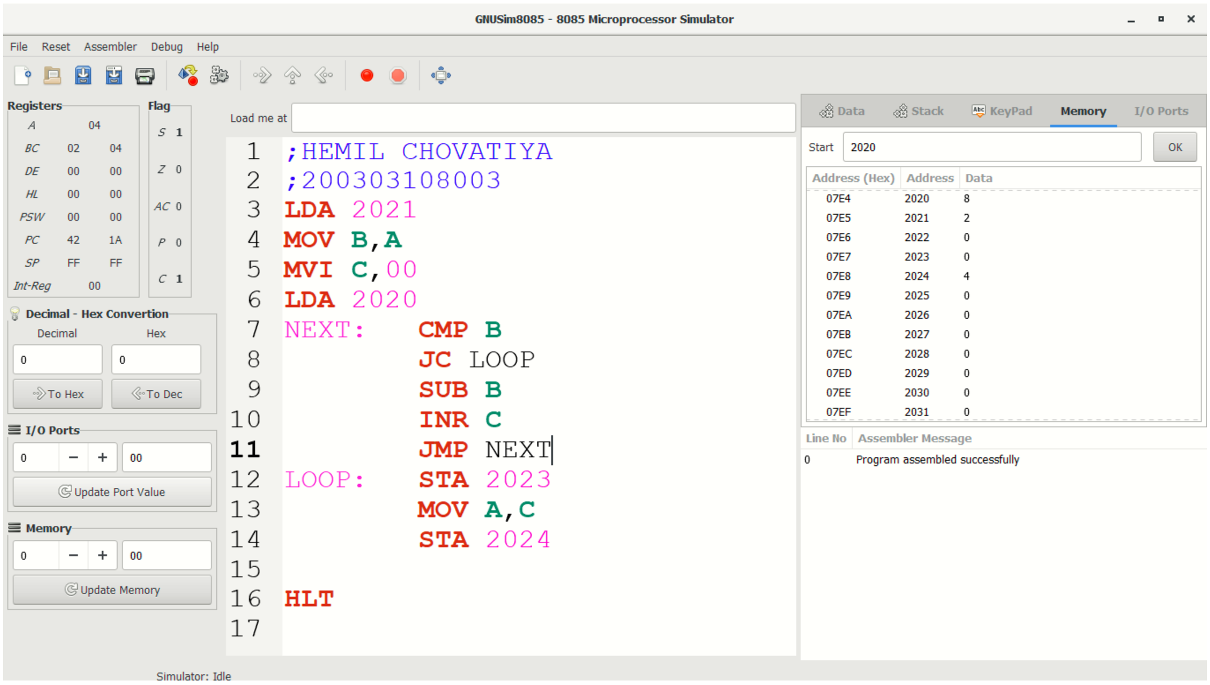
PRACTICAL 8

AIM: **Write an assembly language code in GNUsim8085 to implement Division of two 8bit Numbers**

THEORY

Code	Meaning
LDA 2021	Load value of memory location 2021 in Accumulator A
MOV B, A	Move data from memory to accumulator
MVI C,00	8-bit data is stored in the destination register or memory. (Move immediate 8-bit)
CMP B	Compare register or memorywith accumulator
JC LOOP	Jump if Carry CY=1
SUB B	Subtract register or memoryfrom accumulator
INR C	Increment register or memoryby 1
JMP NEXT	Jump unconditionally (16 Bit Address)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2020=8
2021=2

OUTPUT: 2023=0
2024=4

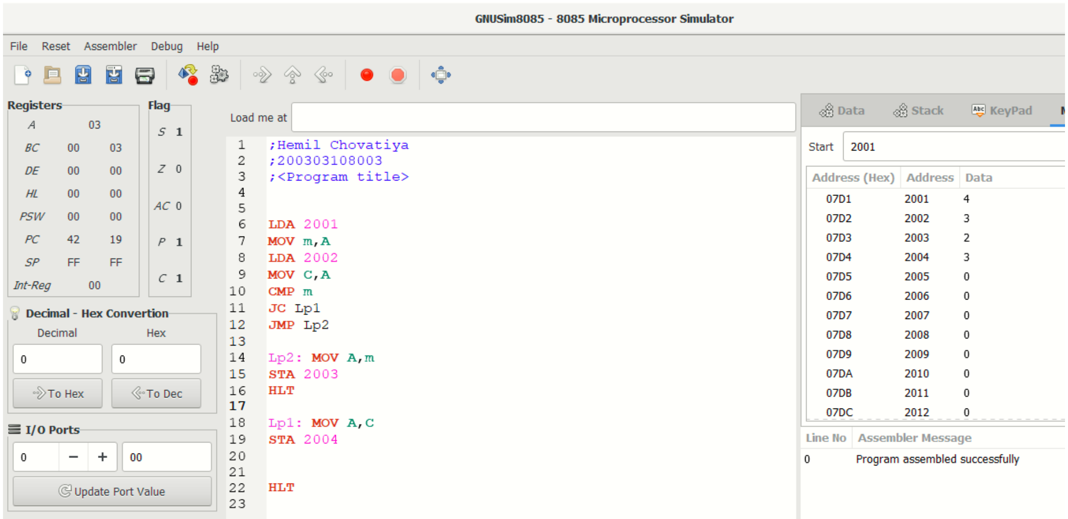
PRACTICAL 9

AIM: **Write an assembly language code in GNUsim8085 to smallest numbers stored in memory**

THEORY

Code	Meaning
LDA 2001	Load value of memory location 2001 in Accumulator A
MOV m, A	Move data from memory to accumulator
CMP B	Compare register or memorywith accumulator
JC Lp1	Jump if Carry CY=1
JMP Lp2	Jump unconditionally (16 Bit Address)
STA 2003	Store accumulator direct(16 bit)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2001=4
2002=3
2003=2

OUTPUT: 2024=3

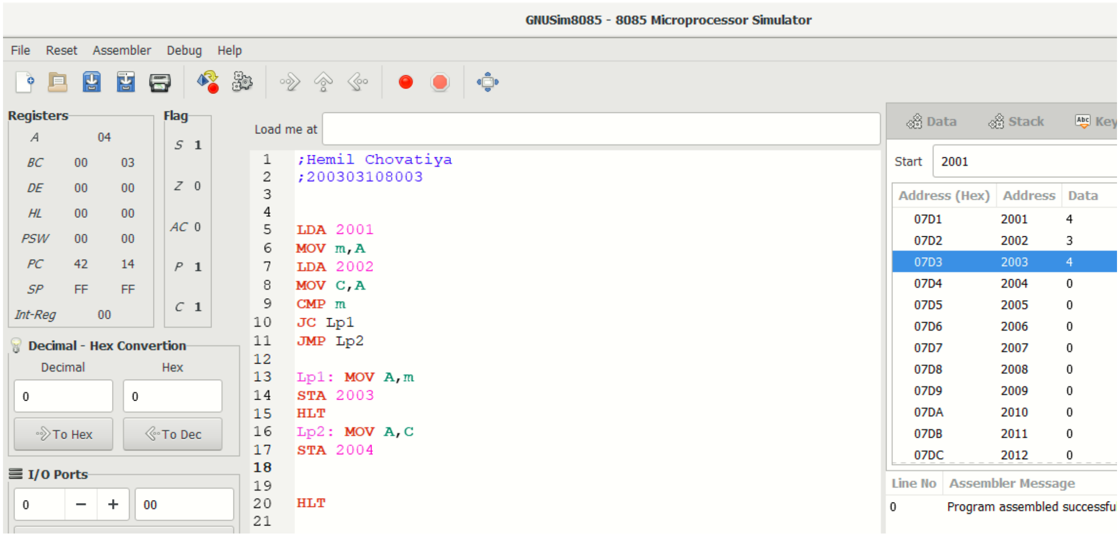
PRACTICAL 10

AIM: Write an assembly language code in GNUsim8085 to larger numbers stored in memory

THEORY

Code	Meaning
LDA 2001	Load value of memory location 2001 in Accumulator A
MOV m, A	Move data from memory to accumulator
CMP B	Compare register or memorywith accumulator
JC Lp1	Jump if Carry CY=1
JMP Lp2	Jump unconditionally (16 Bit Address)
STA 2003	Store accumulator direct(16 bit)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2001=4
2002=3

OUTPUT: 2003=4

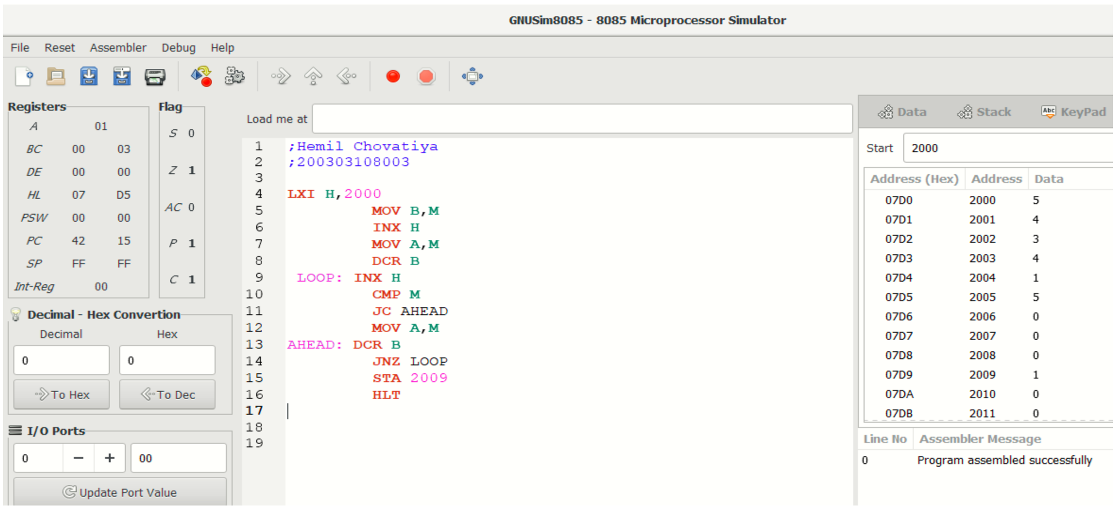
PRACTICAL 11

AIM: Write an assembly language code in GNUsim8085 to smallest numbers among series stored in memory

THEORY

Code	Meaning
LXI H,2000	Load register pair immediate
MOV B, M	Move data from memory to accumulator
INX H	register pair are incremented by 1
DCR B	Decrement register or memory by 1
CMP	Compare register or memory with accumulator
JC AHEAD	Jump if Carry CY=1
JNZ LOOP	Jump if No Zero (Z = 0) to LOOP
STA 2009	Store accumulator direct(16 bit)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2000=5
2001=4
2002=3
2003=4
2004=1
2005=5

OUTPUT: 2009=1

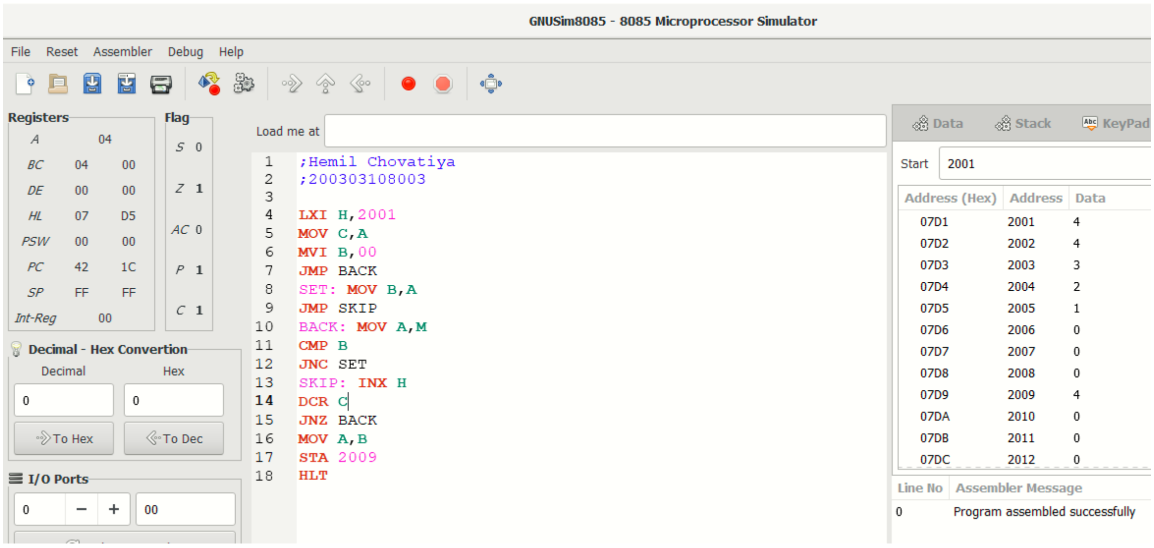
PRACTICAL 12

AIM: Write an assembly language code in GNUsim8085 to Largest numbers among series stored in memory

THEORY

Code	Meaning
LXI H,2001	Load register pair immediate
MOV C, A	Move data from memory to accumulator
MVI B,00	Move immediate 8-bit
JMP SKIP	Jump unconditionally (16 Bit Address)
DCR C	Decrement register or memory by 1
CMP B	Compare register or memory with accumulator
JNC SET	Jump if no carry CY = 0
JNZ BACK	Jump if No Zero (Z = 0) to BACK
STA 2009	Store accumulator direct(16 bit)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2001=4
2002=4
2003=3
2004=2
2005=1

OUTPUT: 2009=4

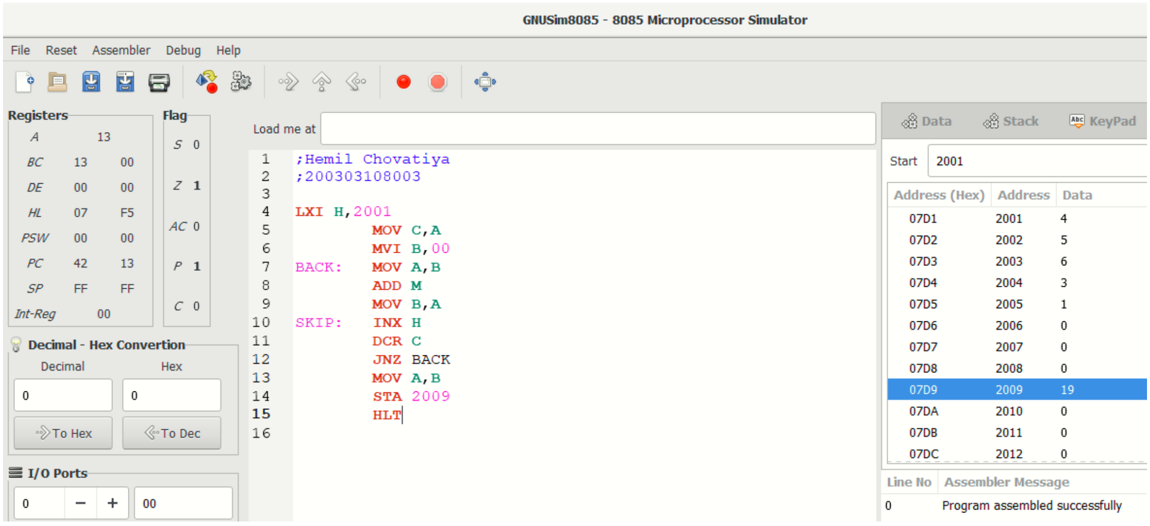
PRACTICAL 13

AIM: Write an assembly language code in GNUsim8085 to Addition of numbers of array stored in memory

THEORY

Code	Meaning
LXI H,2001	Load register pair immediate
MOV C, A	Move data from memory to accumulator
MVI B,00	Move immediate 8-bit
ADD M	Add register or memory to accumulator
INX H	Increment register pair by 1
DCR C	Decrement register or memory by 1
JNZ BACK	Jump if No Zero (Z = 0) to BACK
STA 2009	Store accumulator direct(16 bit)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2001=4
2002=5
2003=6
2004=3
2005=1

OUTPUT: 2009=19

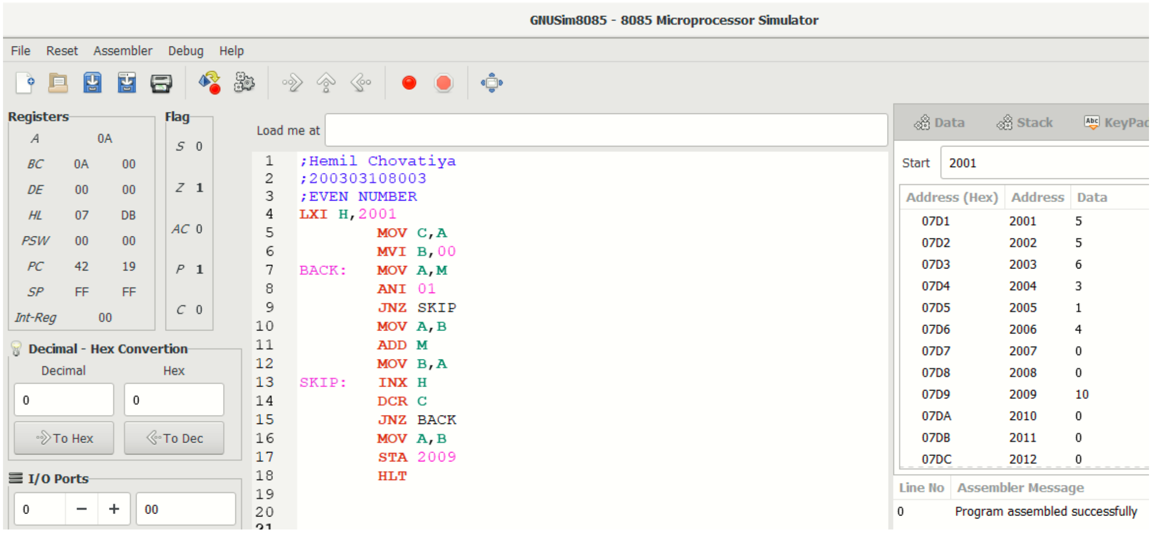
PRACTICAL 14

AIM: Write an assembly language code in GNUsim8085 to even numbers addition stored in memory

THEORY

Code	Meaning
LXI H,2001	Load register pair immediate
MOV C, A	Move data from memory to accumulator
MVI B,00	Move immediate 8-bit
ANI 01	Logical AND immediate with accumulator (CY is reset, AC is set)
JNZ SKIP	Jump if No Zero (Z = 0) to SKIP
ADD M	Add register or memory to accumulator
INX H	Increment register pair by 1
DCR C	Decrement register or memory by 1
STA 2009	Store accumulator direct(16 bit)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2001=5
2002=5
2003=6
2004=3
2005=1
2006=4

OUTPUT: 2009=10

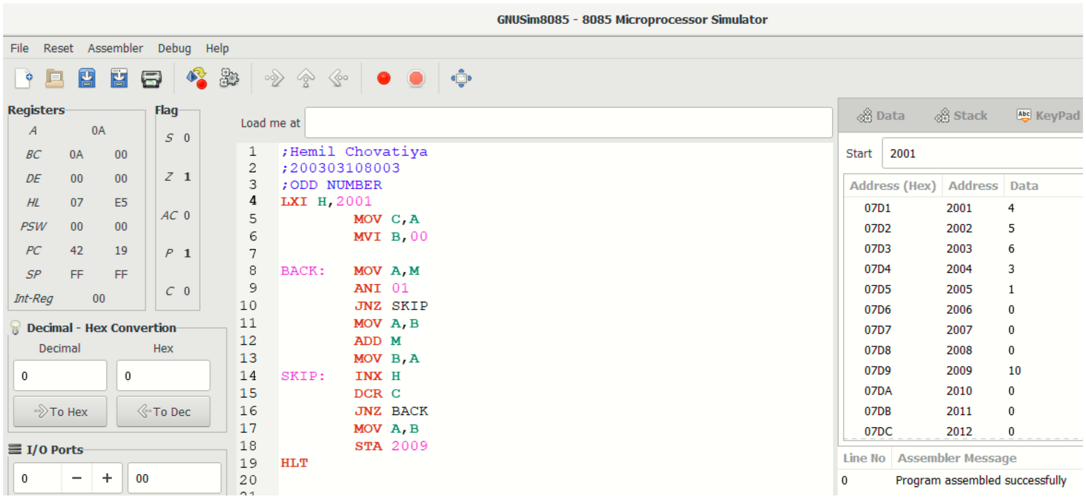
PRACTICAL 15

AIM: Write an assembly language code in GNUsim8085 to Odd numbers addition stored in memory

THEORY

Code	Meaning
LXI H,2001	Load register pair immediate
MOV C, A	Move data from memory to accumulator
MVI B,00	Move immediate 8-bit
ANI 01	Logical AND immediate with accumulator(CY is reset, AC is set)
JNZ SKIP	Jump if No Zero (Z = 0) to SKIP
ADD M	Add register or memory to accumulator
INX H	Increment register pair by 1
DCR C	Decrement register or memory by 1
STA 2009	Store accumulator direct(16 bit)
HLT	Hold the program

IMPLEMENTATION:



INPUT: 2001=4
2002=5
2003=6
2004=3
2005=1

OUTPUT: 2009=10

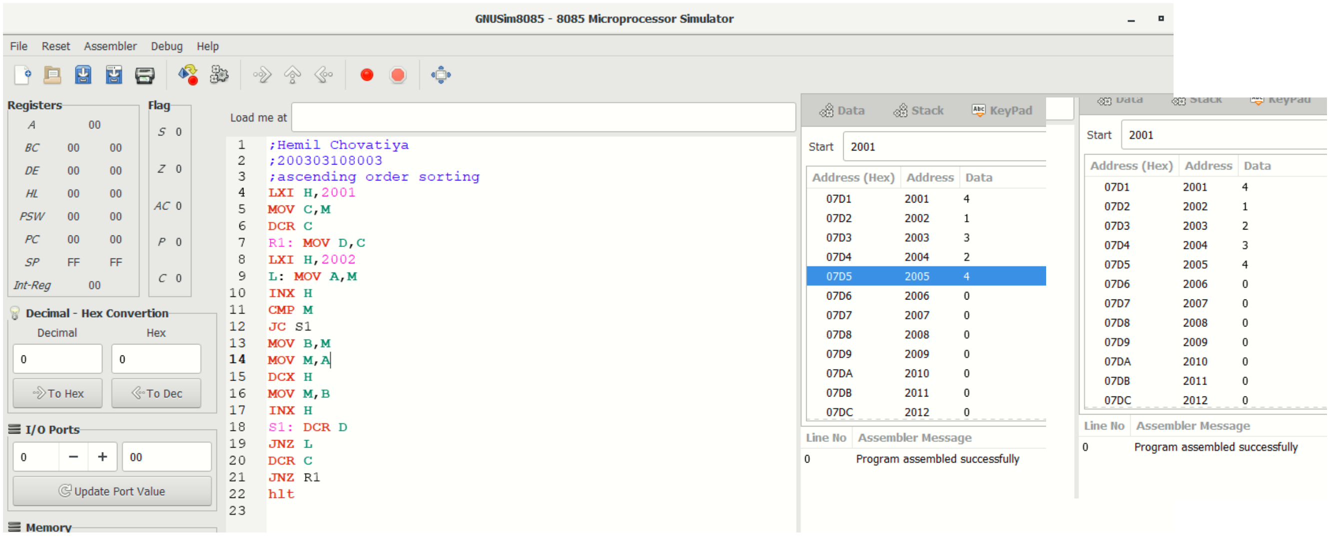
PRACTICAL 16

AIM: Write an assembly language code in GNUsim8085 to arranging numbers in ascending order stored in memory

THEORY

Code	Meaning
LXI H,2001	Load register pair immediate
MOV C, M	Move data from memory to accumulator
DCR C	Decrement register or memory by 1
INX H	Increment register pair by 1
CMP M	Compare register or memory with accumulator
JC S1	Jump if Carry CY=1
DCX H	Decrement register pair by 1
JNZ L	Jump if No Zero (Z = 0) to L
HLT	Hold the program

IMPLEMENTATION:



INPUT:
2001=4
2002=1
2003=3
2004=2
2005=4

OUTPUT:
2001=4
2002=1
2003=2
2004=3
2004=4

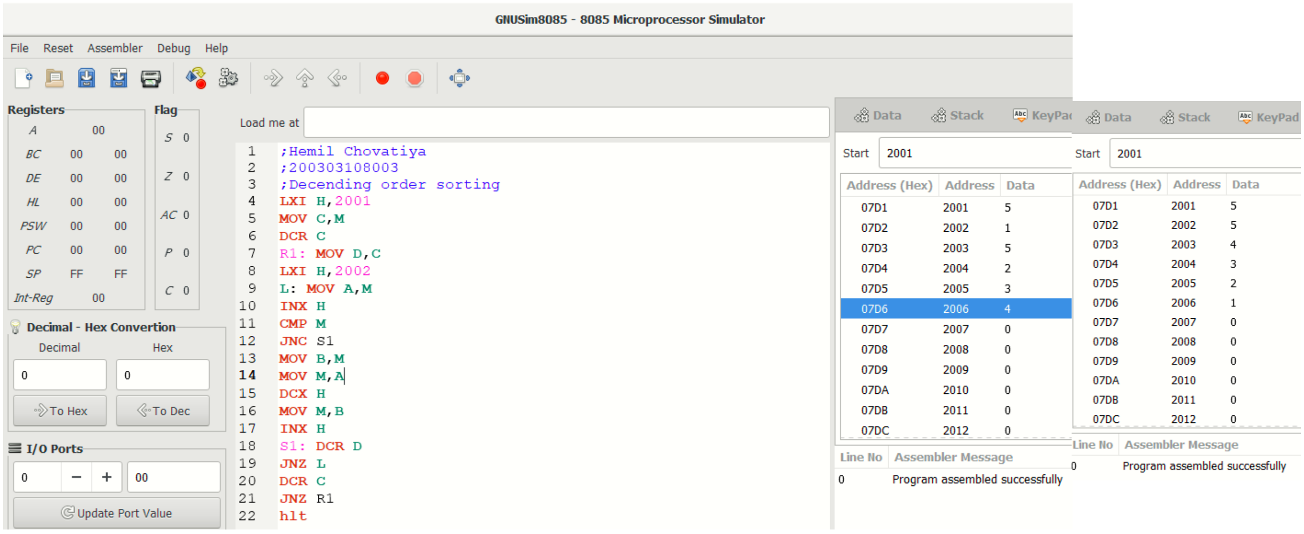
PRACTICAL 17

AIM: Write an assembly language code in GNUsim8085 to arranging numbers in descending order stored in memory

THEORY

Code	Meaning
LXI H,2001	Load register pair immediate
MOV C, M	Move data from memory to accumulator
DCR C	Decrement register or memory by 1
INX H	Increment register pair by 1
CMP M	Compare register or memory with accumulator
JNC S1	Jump if no carry CY = 0
DCX H	Decrement register pair by 1
JNZ L	Jump if No Zero (Z = 0) to L
HLT	Hold the program

IMPLEMENTATION:



INPUT:
2001=5
2002=1
2003=5
2004=2
2005=3
2006=4

OUTPUT:
2001=5
2002=5
2003=4
2004=3
2005=2
2006=1

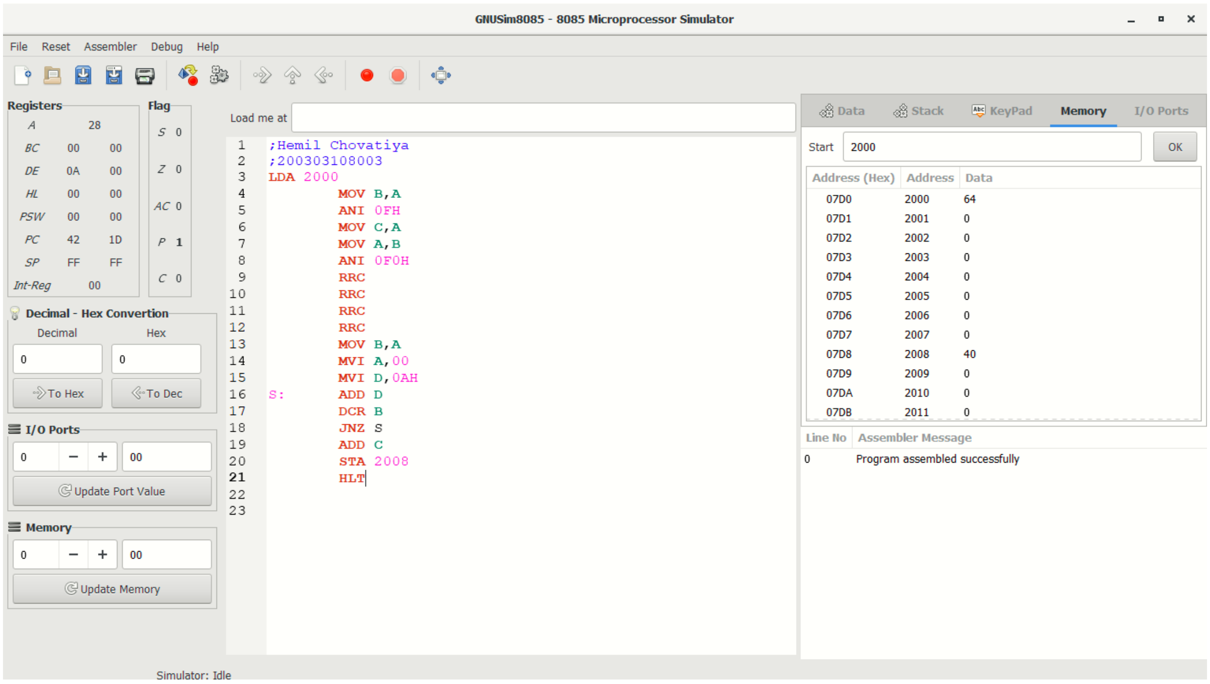
PRACTICAL 18

AIM: Write an assembly language code in GNUsim8085 to convert BCD to binary stored in memory

THEORY

Code	Meaning
LDA 2000H	Load value of memory location 0001 in Accumulator A
MOV C, M	Move data from memory to accumulator
ANI 0fh	Logical AND immediate with accumulator(CY is reset, AC is set)
RRC	Rotate right accumulator(can rotate accumulator current content to right by 1-bit position)
MVI A,00	8-bit data is stored in the destination register or memory. (Move immediate 8-bit)
DCR B	Decrement register or memory by 1
ADD D	Add register or memory to accumulator
JNZ S	Jump if No Zero (Z = 0) to S
HLT	Hold the program

IMPLEMENTATION:



INPUT:

2000=64

OUTPUT:

2008=40