**MICROPROCESSORS BASED SYSTEMS DESIGN**

**(UCS617)**

**LAB ASSIGNMENT**

**Submitted By:**

|  |  |
| --- | --- |
| Gurjot Singh | 102203582 |
| Anil Kumar | 102383026 |
| Simarpreet Singh | 102203606 |
| Samiksha Kak | 102203587 |
| Ananya Bera | 102383024 |

**Group: 3C34**

**BE Third Year**

**Submitted To:**

**Dr. Anju Bala**

(Associate Professor)

Department of Computer Science & Engineering



**Computer Science and Engineering Department**

**Thapar Institute of Engineering and Technology**

**Patiala - 147001**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Description** | **Page No.** |
|  | **8085 Microprocessor** |  |
| 1. | Introduction of 8085-microprocessor kit and steps for execution on the kit. | 1 – 9 |
| 2. | Familiarity with 8085-microprocessor kit. |  |
| i) | Write a program to store 8-bit data into one register and then copy that to all registers. | 10 – 11 |
| ii) | Write a program for addition of two 8-bit numbers | 12 – 13 |
| iii) | Write a program to add 8-bit numbers using direct and indirect addressing mode | 14 - 16 |
| iv) | Write a program to add 16-bit numbers using direct and indirect addressing mode. | 17 – 20 |
| v) | Write a program to 8-bit numbers using carry. (using JNC instruction). | 21 – 22 |
| vi) | Write a program to find 1’s complement and 2’s complement of 8-bit number. | 23 – 26 |
| 3. | Write a program for the sum of series of numbers. | 27 - 28 |
| 4. | Write a program for data transfer from memory block B1 to memory block B2. | 29-30 |
| 5. | Write a program for data transfer from memory block B1 to memory block B2. | 31-32 |
| 6. | Write a program to add ten 8-bit numbers. Assume the numbers are stored in 8500-8509. Store the result in 850A and 850B memory address. | 33-34 |
| 7. | Write a program to find the negative numbers in a block of data. | 35-36 |
| 8. | Write a program to count the number of one's in a number. | 37-38 |
| 9. | Write a program to arrange numbers in Ascending order. | 39-41 |
| 10. | Calculate the sum of series of even numbers. | 42-43 |
| 11. | Write an assembly language program to verify how many bytes are present in a given set, which resembles 10101101 in 8085 | 44-45 |
| 12. | Write an assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085. | 46-47 |
| 13. | Write an assembly language program to convert a BCD number into its equivalent binary in 8085. | 48-49 |
| 14. | Write an assembly language program for exchange the contents of memory location. | 50-51 |
| 15. | Write a program to find the largest number in an array of 10 elements. | 52-53 |

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Description** | **Page No.** |
|  | **8086 Microprocessor** |  |
| 16. | Write an assembly language program to add two 16-bit numbers in 8086. | 3 |
| 17. | Write an assembly language program to subtract two 16-bit numbers in 8086. | 4 |
| 18. | Write an assembly language program to multiply two 16-bit numbers in 8086. | 5 |
| 19. | Write an assembly language program to divide two 16-bit numbers in 8086. | 6 |
| 20. | Write an assembly language program to demonstrate AAA, AAS, AAM, AAD, DAA and DAS in 8086. | 7-12 |
| 21. | Write an assembly language program to find out the count of positive numbers and negative numbers from a series of signed numbers in 8086. | 13 |
| 22. | Write an assembly language program to convert to find out the largest number from a given unordered array of 8-bit numbers, stored in the locations starting from a known address in 8086. | 14 |
| 23. | Write an assembly language program to convert to find out the largest number from a given unordered array of 16-bit numbers, stored in the locations starting from a known address in 8086. | 15 |
| 24. | Write an assembly language program to print Fibonacci series in 8086. | 16 |
| 25. | Write an assembly language program to perform the division 15/6 using the ASCII codes. Store the ASCII codes of the result in register DX. | 17 |
|  | **Interfacing using 8086-Microprocessor Kit** |  |
| 26. | Interfacing ET-8255 study card with ET-8086 microprocessor hardware kit. |  |
| 27. | Interfacing ET-8253 study card with ET-8086 microprocessor hardware kit. |  |
| 28. | Interfacing ET-8279 study card with ET-8086 microprocessor hardware kit. |  |
| 29. | Interfacing ET-8251 study card with ET-8086 microprocessor hardware kit. |  |
| 30. | Interfacing ET-8259 study card with ET-8086 microprocessor hardware kit. |  |

**QUESTION NO: 1**

**OBJECTIVE:**

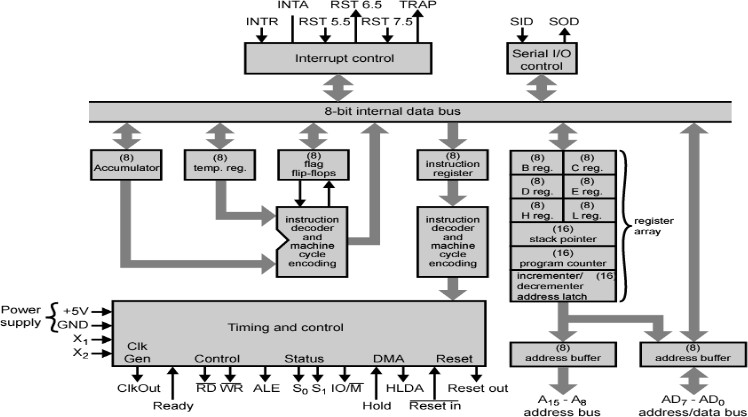
Introduction of 8085-microprocessor kit and steps for execution on the kit.

The Intel 8085 microprocessor is an 8-bit microprocessor introduced by Intel in 1976. It is part of the 8080-microprocessor family and is often considered an enhanced version of the Intel 8080. The 8085 microprocessor was widely used in early microcomputer systems and embedded applications. Key features of the Intel 8085 microprocessor include:

* **ARCHITECTURE:** 8-bit data bus, 16-bit address bus, addressing up to 64 KB memory, single accumulator for arithmetic/logic operations.
* **REGISTERS:** Accumulator (ACC), general-purpose (B, C, D, E, H, L), special- purpose (SP, PC).
* **CLOCK SPEED:** Operates at 3 MHz (varying in different versions).
* **INSTRUCTION SET:** Based on 8080, enhanced with additional instructions, supports data manipulation, control, and I/O operations.
* **ADDRESSING MODES:** Immediate, direct, indirect, and register addressing modes.
* **INTERRUPTS:** Supports TRAP, RST 7.5, RST 6.5, RST 5.5, INTR interrupt lines.
* **PERIPHERAL INTERFACE:** Interfaces with peripherals through I/O instructions and ports.
* **POWER SUPPLY:** Typically operates with a +5V power supply.



**Fig. 1.1:** Intel 8085 Microprocessor Training Kit



**Fig. 1.2:** Block Diagram of Intel 8085 Microprocessor

1. **Address Bus (Pin 21-28)**
   1. 16 bit address lines A0 to A15.
   2. The address bus has 8 signal lines A8 – A15 which are unidirectional.
   3. The other 8 address lines A0 to A7 are multiplexed (time shared) with the 8 data bits.
2. **Data Bus (Pin 19-12)**
   1. To save the number of pins lower order address pin are multiplexed with 8 bit data bus (bidirectional)
   2. So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
   3. During the execution of the instruction, these lines carry the address bits during the early part (T1 state), then during the late parts (T2 state) of the execution, they carry the 8 data bits.
3. **Status Pins – ALE, S1, S0**
   1. **ALE(Address Latch Enable) : (Pin 30)**
      1. Used to demultiplexed the address and data bus.

ii. +ve going pulse generated when a new operation is started by microprocessor.

1. ALE = 1 when the AD0 – AD7 lines have an address.
2. ALE = 0 When it is low it indicates that the contents are data.
3. This signal can be used to enable a latch to save the address bits from the AD lines.
   1. **S1 and S0 (Status Signal) : (Pin 33 and 29)**
      1. Status signals to specify the kind of operation being performed .
      2. Usually un-used in small systems.

|  |  |  |
| --- | --- | --- |
| **S1** | **So** | **Operations** |
| 0 | 0 | HALT |
| 0 | 1 | WRITE |
| 1 | 0 | READ |
| 1 | 1 | FETCH |

**Table 1.1:** Status Signals

1. **Control Pins – RD, WR, IO/M (active low)**
   1. **RD: Read (Active low) (Pin 32)**
      1. Read Memory or I/O device.
      2. Indicated that data is to be read either from memory or I/P device and data bus is ready for accepting data from the memory or I/O device.
   2. **WR: Write (Active low) (Pin 31)**
      1. Write Memory or I/O device
      2. Indicated that data on the data bus are to be written into selected memory or I/P device.
   3. **IO/M: (Input Output/Memory-Active low) (Pin 34)**
      1. Signal specifies that the read/write operation relates to whether memory or I/O device.
      2. When (IO/M=1) the address on the address bus is for I/O device.
      3. When (IO/M=0) the address on the address bus is for memory.

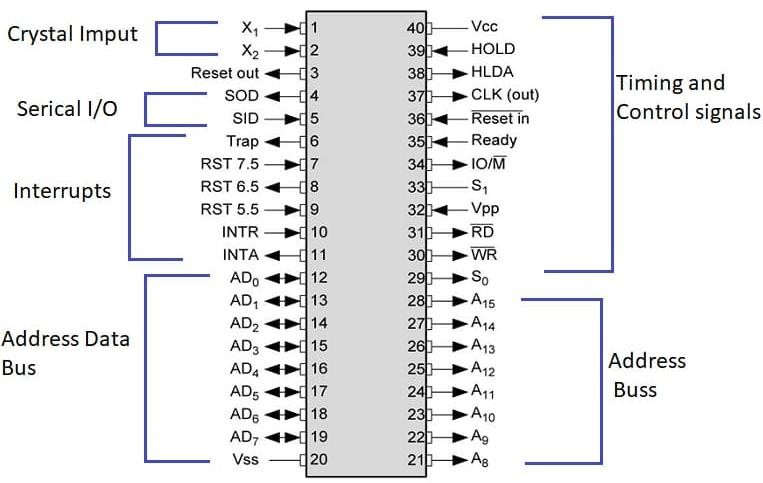
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IO/M(active low)** | **RD** | **WR** | **Control Signals** | **Operations** |
| 0 | 0 | 1 | MEMR | M/M Read |
| 0 | 1 | 0 | MEMW | M/M Write |
| 1 | 0 | 1 | IOR | I/O Read |
| 1 | 1 | 0 | IOW | I/O Write |

**Table 1.2:** Control Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IO/M** | **S1** | **S0** | **Operations** | **Control Signals** |
| 0 | 1 | 1 | Opcode Fetch | 𝑅̅̅𝐷̅̅=0 |
| 0 | 1 | 0 | Memory Read | 𝑅̅̅𝐷̅̅=0 |
| 0 | 0 | 1 | Memory Write | 𝑊̅̅̅𝑅̅̅=0 |
| 1 | 1 | 0 | I/O Read | 𝑅̅̅𝐷̅̅=0 |
| 1 | 0 | 1 | I/O Write | 𝑊̅̅̅𝑅̅̅=0 |
| 1 | 1 | 0 | Interrupt Acknowledge | 𝐼̅𝑁̅̅𝑇̅̅𝐴̅̅=0 |
| Z | 0 | 0 | Halt | 𝑅̅̅𝐷̅̅, 𝑊̅̅̅𝑅̅̅=Z & 𝐼̅𝑁̅̅𝑇̅̅𝐴̅̅=1 |
| Z | X | X | Hold |
| Z | X | X | Reset |

**Table 1.3:** Control & Status Signals

1. **Interrupt Pins (Pin 6-11)**
   1. They are the signals initiated by an external device to request the microprocessor to do a particular task or work.
   2. On receipt of an interrupt, the microprocessor acknowledges the interrupt by the active low INTA (Interrupt Acknowledge) signal.
2. **Power Supply & Clock Signal**
   1. **Vcc : (Pin 40)**
      1. Single +5 volt power supply
   2. **Vss : (Pin 20)**
      1. Ground.
   3. **X0 and X1 : (Pin 1-2)**
      1. Crystal or R/C network or LC network connections to set the frequency of internal clock generator.
      2. The frequency is internally divided by two.
      3. Since the basic operating timing frequency is 3 MHz, a 6 MHz crystal is connected to the X0 and X1 pins.
   4. **CLK (output) : (Pin 37)**
      1. Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.
3. **HOLD (Pin 38)**
   1. This signal indicates that another device is requesting the use of address and data bus.
   2. So it relinquish the use of buses as soon as the current machine cycle is completed.
   3. Microprocessor regains the bus after the removal of a HOLD signal.
4. **HLDA (Pin 39)**
   1. On receipt of HOLD signal, the MP acknowledges the request by sending out HLDA signal and leaves out the control of the buses.
   2. After the HLDA signal the DMA controller starts the direct transfer of data.
   3. After the removal of HOLD request HLDA goes low.
5. **SID (input) Serial input data (Pin 4)**
   1. It is a data line for serial input.
   2. Used to accept serial data bit by bit from external device.
   3. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
6. **SOD (output) Serial output data (Pin 5)**
   1. It is a data line for serial output.
   2. Used to transmit serial data bit by bit to the external device.
   3. The 7th bit of the accumulator is outputted on SOD line when SIM instruction is executed.
7. **Ready (input) (Pin 35)**
   1. Memory and I/O devices will have slower response compared to microprocessors.
   2. Before completing the present job such a slow peripheral may not be able to handle further data or control signals from CPU.
   3. The processor sets the READY signal after completing the present job to access the data.
   4. It synchronize slower peripheral to the processor.

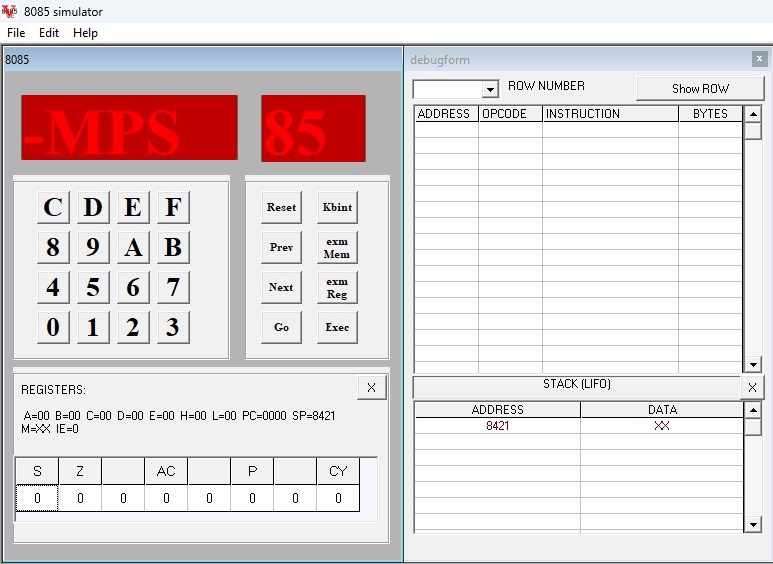


**Fig. 1.3:** Pin Diagram of Intel 8085 Microprocessor

**FEATURES of MICROPROCESSOR – 8085**

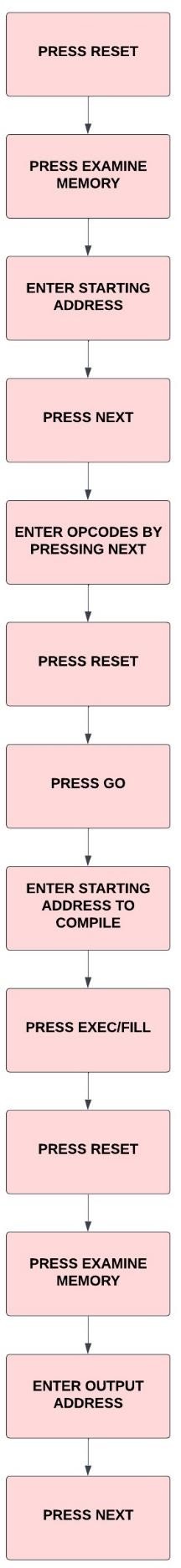
1. 8085 is developed by INTEL.
2. 8-bit microprocessor: can accept 8-bit data simultaneously.
3. Operates on single +5V D.C. supply.
4. Designed using NMOS technology.
5. 6200 transistors on a single chip.
6. It provides on chip clock generator; hence it does not require external clock generator.
7. Operates on 3MHz clock frequency.
8. 8bit multiplexed address/data bus, which reduce the number of pins.
9. 16address lines, hence it can address 2^16 = 64 K bytes of memory
10. It generates 8 bit I/O addresses; hence it can access 2^8 = 256 1/0 ports. 5 hardware interrupts i.e. TRAP/RST4.5, RST 7.5, RST 6.5, RST 5.5, and INTR
11. It provides DMA (Direct memory access).
12. 40-pin L.C. package fabricated on a single LSI chip.
13. Clock cycle is 320ns.
14. 80 basic instructions and 246 opcodes.

**VIKAS SIMULATOR**

****

**Fig. 1.4:** Vikas Stimulator

**STEPS TO PERFORM ON VIKAS SIMULATOR**

****

**Fig. 1.5:** Steps to perform on Vikas Stimulator

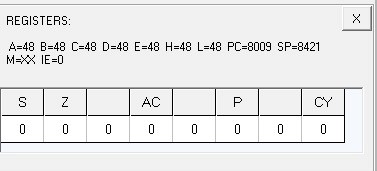
**QUESTION NO: 2(i)**

**OBJECTIVE:**

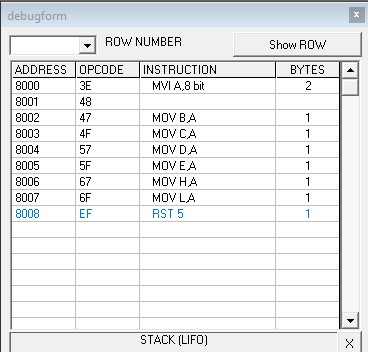
Write a program to store 8-bit data into one register and then copy that to all registers.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| MVI A, 48 | 8000, 8001 | 3E, 48 |
| MOV B, A | 8002 | 47 |
| MOV C, A | 8003 | 4F |
| MOV D, A | 8004 | 57 |
| MOV E, A | 8005 | 5F |
| MOV H, A | 8006 | 67 |
| MOV L, A | 8007 | 6F |
| RST 5 | 8008 | EF |

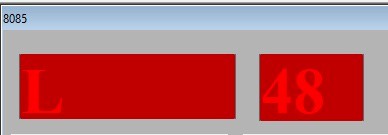
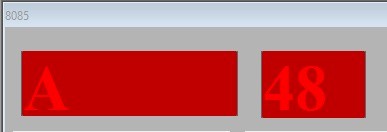
**Table 2(i).1:** Code Explanation



**Fig. 2(i).1:** Register Output



**Fig. 2(i).2:** Debugform



**Fig. 2(i).3:** Output

**OUTPUT –**

A – 48, B – 48, C – 48, D – 48, E – 48, H – 48, L – 48

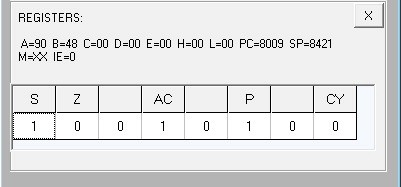
**QUESTION NO: 2(ii)**

**OBJECTIVE:**

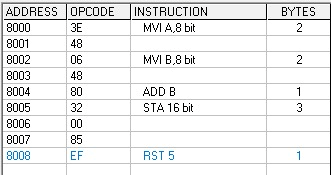
Write a program for addition of two 8-bit numbers.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| MVI A, 48 | 8000, 8001 | 3E, 48 |
| MVI B, 48 | 8002, 8003 | 06, 48 |
| ADD B | 8004 | 80 |
| STA 8500 | 8005, 8006, 8007 | 32, 00, 85 |
| RST 5 | 8008 | EF |

**Table 2(ii).1:** Code Explanation



**Fig. 2(ii).1:** Register Output



**Fig. 2(ii).2:** Debugform



**Fig. 2(ii).3:** Output

**OUTPUT –**

[ 8500] – 90

**QUESTION NO: 2(iii)**

**OBJECTIVE:**

Write a program to add 8-bit numbers using direct and indirect addressing mode.

1. **Direct Addressing**

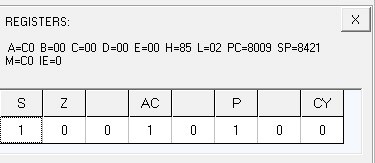
|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| LDA 8500 | 8000, 8001, 8002 | 3A, 00, 85 |
| MOV B, A | 8003 | 47 |
| LDA 8501 | 8004, 8005, 8006 | 3A, 01, 85 |
| ADD B | 8007 | 80 |
| STA 8502 | 8008, 8009, 800A | 32, 02, 85 |
| RST 5 | 800B | EF |

**Table 2(iii).1:** Code Explanation

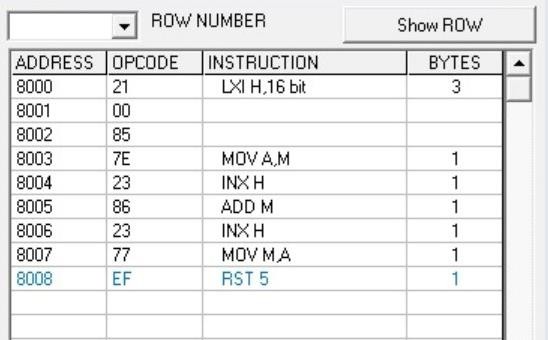
1. **In-Direct Addressing**

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| LXI H, 8500 | 8000, 8001, 8002 | 21, 00, 85 |
| MOV A, M | 8003 | 7E |
| INX H | 8004 | 23 |
| ADD M | 8005 | 86 |
| INX H | 8006 | 23 |
| MOV M, A | 8007 | 77 |
| RST 5 | 8008 | EF |

**Table 2(iii).2:** Code Explanation



**Fig. 2(iii).1:** Register Output



**Fig. 2(iii).2:** Debugform



**Fig. 2(iii).3:** Output

**INPUT –**

[ 8500] – 58, [ 8501] – 68

**OUTPUT –**

[ 8502] – C0

**QUESTION NO: 2(iv)**

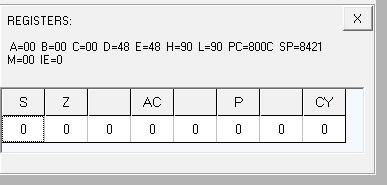
**OBJECTIVE:**

Write a program to add 16-bit numbers using direct and indirect addressing mode.

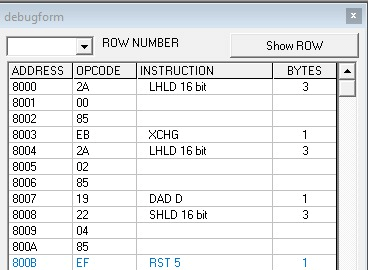
1. **Direct Addressing**

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| LHLD 8500 | 8000, 8001, 8002 | 2A, 00, 85 |
| XCHG | 8003 | EB |
| LHLD 8502 | 8004, 8005, 8006 | 2A, 02, 85 |
| DAD D | 8007 | 19 |
| SHLD 8504 | 8008, 8009, 800A | 22, 04, 85 |
| RST 5 | 800B | EF |

**Table 2(iv).1:** Code Explanation



**Fig 2(iv).1:** Register Output



**Fig 2(iv).2:** Debugform



**Fig 2(iv).3:** Output

**INPUT –**

[ 8500] – 48, [ 8501] – 48, [ 8502] – 48, [ 8503] – 48

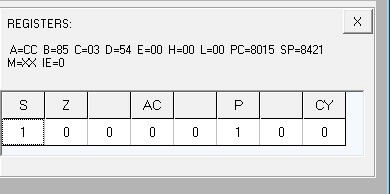
**OUTPUT –**

[ 8504] – 90, [ 8505] – 90

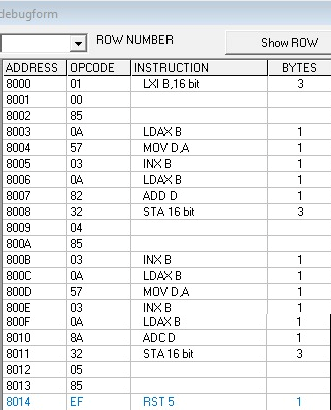
1. **In-Direct Addressing**

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| LXI B, 8500 | 8000, 8001, 8002 | 01, 00, 85 |
| LDAX B | 8003 | 0A |
| MOV D, A | 8004 | 57 |
| INX B | 8005 | 03 |
| LDAX B | 8006 | 0A |
| ADD D | 8007 | 82 |
| STA 8504 | 8008, 8009, 800A | 32, 04, 85 |
| INX B | 800B | 03 |
| LDAX B | 800C | 0A |
| MOV D, A | 800D | 57 |
| INX B | 800E | 03 |
| LDAX B | 800F | 0A |
| ADC D | 8010 | 8A |
| STA 8505 | 8011, 8012, 8013 | 32, 05, 85 |
| RST 5 | 8014 | EF |

**Table 2(iv).2:** Code Explanation



**Fig. 2(iv).4:** Register Output



**Fig. 2(iv).5:** Debugform



**Fig. 2(iv).6:** Output

**INPUT –**

[ 8500] – 34, [ 8501] – 48, [ 8502] – 54, [ 8503] – 78

**OUTPUT –**

[ 8504] – 7C [ 8505] – CC

**QUESTION NO: 2(v)**

**OBJECTIVE:**

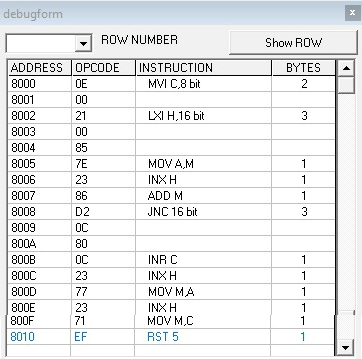
Write a program to 8-bit numbers using carry. (using JNC instruction).

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| MVI C, 00 | 8000,8001 | 0E,00 |
| LXI H, 8500 | 8002,8003,8004 | 21,00,85 |
| MOV A, M | 8005 | 7E |
| INX H | 8006 | 23 |
| ADD M | 8007 | 86 |
| JNC Next | 8008,8009,800A | D2,0C,80 |
| INR C | 800B | 0C |
| Next: INX H | 800C | 23 |
| MOV M, A | 800D | 77 |
| INX H | 800E | 23 |
| MOV M, C | 800F | 71 |
| RST 5 | 8010 | EF |

**Table 2(v).1:** Code Explanation



**Fig. 2(v).1:** Register Output



**Fig. 2(v).2:** Debugform



**Fig. 2(v).3:** Output

**INPUT –**

[ 8500] – 88, [ 8501] – 88

**OUTPUT –**

[ 8502] – 10, [ 8503] – 01

**QUESTION NO: 2(vi)**

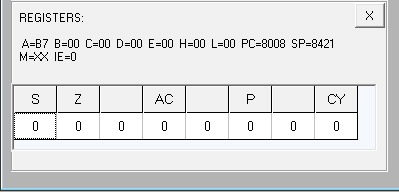
**OBJECTIVE:**

Write a program to find 1’s complement and 2’s complement of 8-bit number.

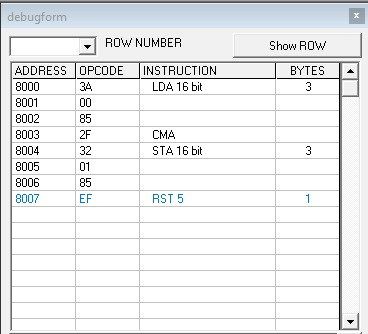
1. **1’s Compliment**

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| LDA 8500H | 8000, 8001, 8002 | 3A, 00, 85 |
| CMA | 8003 | 2F |
| STA 8501H | 8004,8005,8006 | 32, 01, 85 |
| RST 5 | 8007 | EF |

**Table 2(vi).1:** Code Explanation



**Fig 2(vi).1:** Register Output



**Fig 2(vi).2:** Debugform



**Fig 2(vi).3:** Output

**INPUT –**

[ 8500] – 48

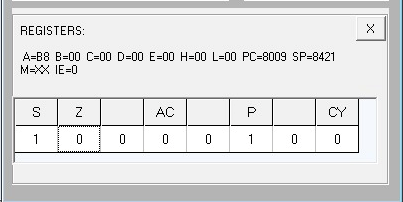
**OUTPUT –**

[ 8501] – B7

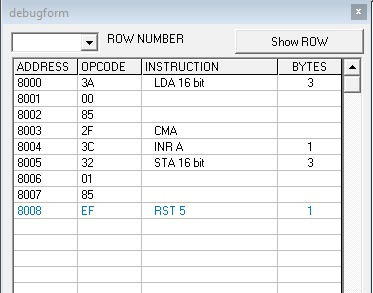
1. **2’s Compliment**

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | 8000, 8001, 8002 | 3A, 00, 85 |
| **CMA** | 8003 | 2F |
| **INR A** | 8004 | 3C |
| **STA 8501H** | 8005,8006,8007 | 32, 01, 85 |
| **RST 5** | 8008 | EF |

**Table 2(vi).2:** Code Explanation



**Fig. 2(vi).4:** Register Output



**Fig. 2(vi).5:** Debugform



**Fig. 2(vi).6:** Output

**INPUT –**

[ 8500] – 48

**OUTPUT –**

[ 8501] – B8

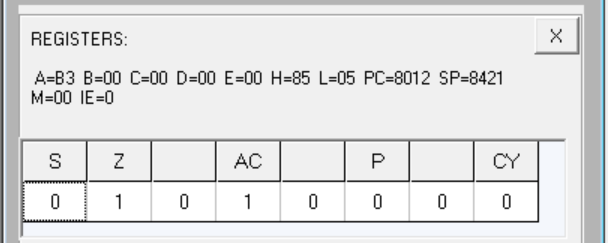
**QUESTION NO: 3**

**OBJECTIVE:**

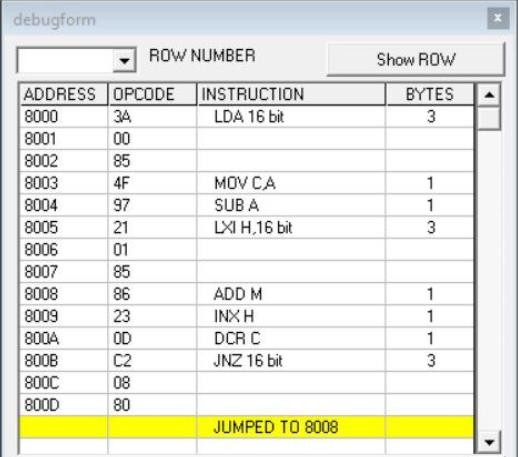
Write a program for the sum of series of numbers.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | 8000,8001,8002 | 3A,00,85 |
| **MOV C, A** | 8003 | 4F |
| **SUB A** | 8004 | 97 |
| **LXI H, 8501H** | 8005,8006,8007 | 21,01,85 |
| **Back: ADD M** | 8008 | 86 |
| **INX H** | 8009 | 23 |
| **DCR C** | 800A | 0D |
| **JNZ Back** | 800B,800C,800D | C2,08,80 |
| **STA 8600H** | 800E,800F,8010 | 32,00,86 |
| **RST 5** | 8011 | EF |

**Table 3.1:** Code Explanation



**Fig. 3.1:** Register Output



**Fig. 3.2:** Debugform



**Fig. 3.3:** Output

**INPUT –**

[8500] – 04, [8501] – 9A, [8502] – 52, [8503] – 89, [8504] – 3E

**RESULT –**

1B3

**OUTPUT –**

B3

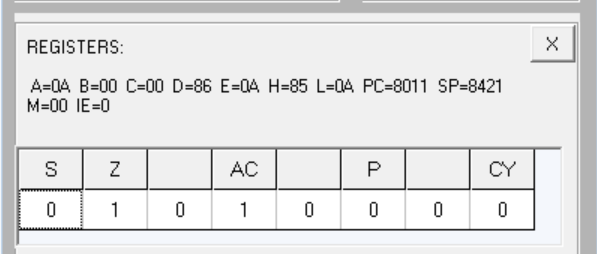
**QUESTION NO: 4**

**OBJECTIVE:**

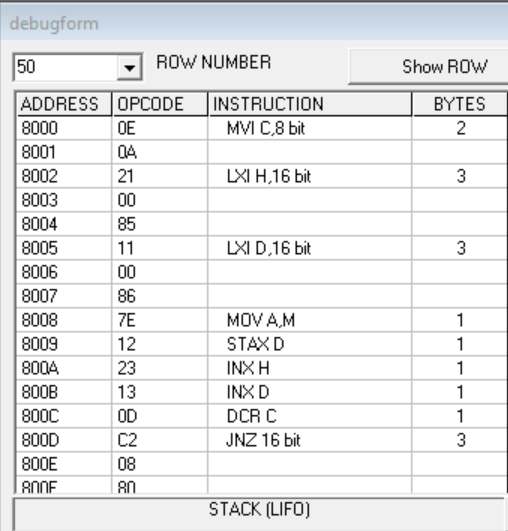
Write a program for data transfer from memory block B1 to memory block B2.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **MVI C, 0AH** | 8000,8001 | 0E,0A |
| **LXI H, 8500H** | 8002,8003,8004 | 21,00,85 |
| **LXI D, 8600H** | 8005,8006,8007 | 11,00,86 |
| **Back: MOV A, M** | 8008 | 7E |
| **STAX D** | 8009 | 12 |
| **INX H** | 800A | 23 |
| **INX D** | 800B | 13 |
| **DCR C** | 800C | 0D |
| **JNZ Back** | 800D,800E,800F | C2,08,80 |
| **RST 5** | 8010 | EF |

**Table 4.1:** Code Explanation



**Fig. 4.1:** Register Output



**Fig. 4.2:** Debugform



…………………………………………………………………………………

……………………………...

**Fig. 4.3:** Output

**INPUT –**

[8500] – 01, [8501] – 02, [8502] – 03……… [8509] – 0A

**OUTPUT –**

[8600] – 01, [8601] – 02, [8602] – 03… [8609] – 0A

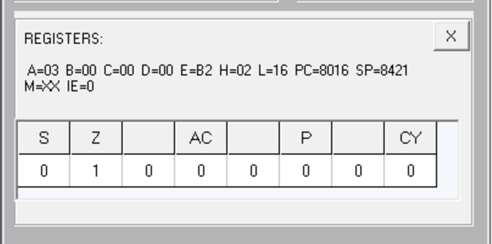
**QUESTION NO: 5**

**OBJECTIVE:**

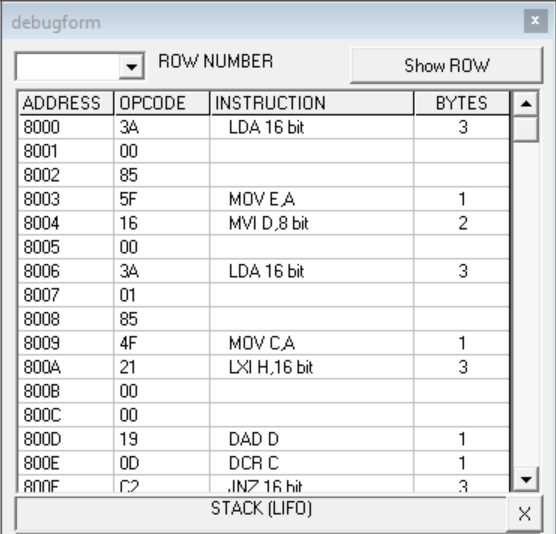
Write a program for multiply two 8-bit numbers.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | 8000,8001,8002 | 3A,00,85 |
| **MOV E, A** | 8003 | 5F |
| **MVI D, 00** | 8004,8005 | 16,00 |
| **LDA 8501H** | 8006,8007,8008 | 3A,01,85 |
| **MOV C, A** | 8009 | 4F |
| **LXI H, 0000H** | 800A,800B,800C | 21,00,00 |
| **Back: DAD D** | 800D | 19 |
| **DCR C** | 800E | 0D |
| **JNZ Back** | 800F,8010,8011 | C2,0D,80 |
| **SHLD 8600H** | 8012,8013,8014 | 22,00,86 |
| **RST 5** | 8015 | EF |

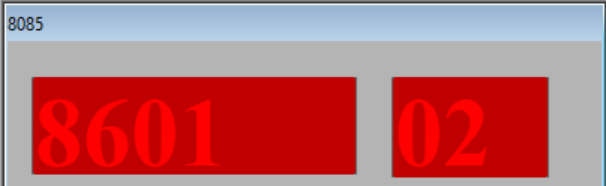
**Table 5.1:** Code Explanation



**Fig. 5.1:** Register Output



**Fig. 5.2:** Debugform



**Fig. 5.3:** Output

**INPUT –**

[8500] – B2, [8501] – 03

**RESULT–**

B2 + B2 + B2 = 0216 H

**OUTPUT –**

[8600] – 16, [8601] – 02

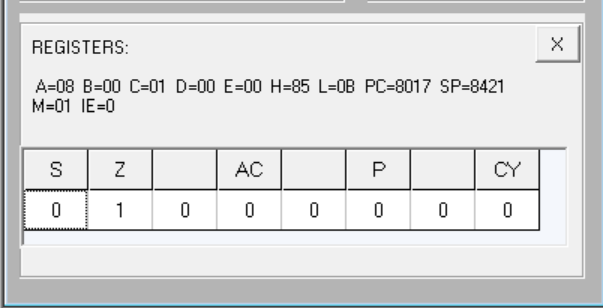
**QUESTION NO: 6**

**OBJECTIVE:**

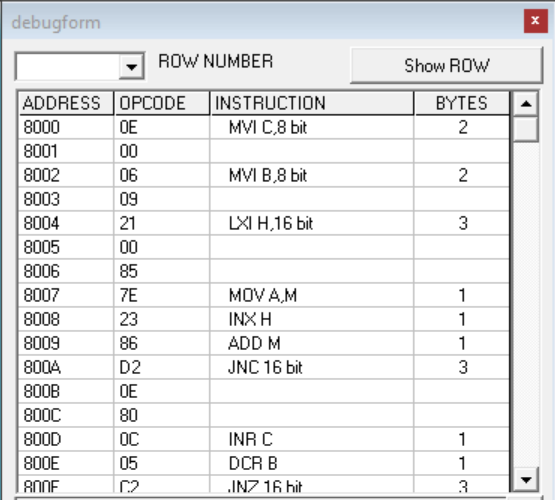
Write a program to add ten 8-bit numbers. Assume the numbers are stored in 8500-8509. Store the result in 850A and 850B memory address.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **MVI C, 00** | 8000,8001 | 0E, 00 |
| **MVI B, 09** | 8002,8003 | 06, 09 |
| **LXI H, 8500H** | 8004,8005,8006 | 21,00,85 |
| **MOV A, M** | 8007 | 7E |
| **Back: INX H** | 8008 | 23 |
| **ADD M** | 8009 | 86 |
| **JNC Next** | 800A,800B,800C | D2, 0E,80 |
| **INR C** | 800D | 0C |
| **Next: DCR B** | 800E | 05 |
| **JNZ Back** | 800F,8010,8011 | C2,08,80 |
| **INX H** | 8012 | 23 |
| **MOV M, A** | 8013 | 77 |
| **INX H** | 8014 | 23 |
| **MOV M, C** | 8015 | 71 |
| **RST 5** | 8016 | EF |

**Table 6.1:** Code Explanation



**Fig. 6.1:** Register Output



**Fig. 6.2:** Debugform



**Fig. 6.3:** Output

**INPUT –**

[8500] – FF, [8501] – 01, [8502] – 01, [8503] – 01, [8504] – 01, [8505] – 01, [8506] – 01,

[8507] – 01, [8508] – 01, [8509] – 01

**OUTPUT –**

[850A] – 08, [850B] – 01

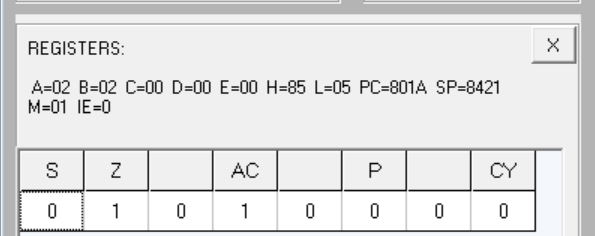
**QUESTION NO: 7**

**OBJECTIVE:**

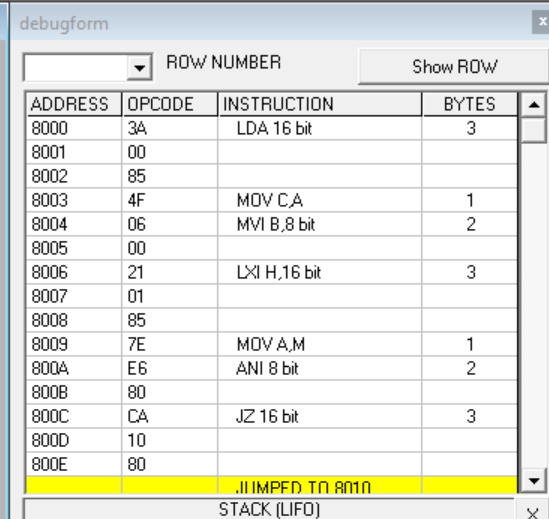
Write a program to find the negative numbers in a block of data.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | 8000,8001,8002 | 3A,00,85 |
| **MOV C, A** | 8003 | 4F |
| **MVI B, 00** | 8004,8005 | 06, 00 |
| **LXI H, 8501H** | 8006,8007,8008 | 21,01,85 |
| **Back: MOV A, M** | 8009 | 7E |
| **ANI 80H** | 800A,800B | E6,80 |
| **JZ Skip** | 800C,800D,800E | CA,10,80 |
| **INR B** | 800F | 04 |
| **Skip: INX H** | 8010 | 23 |
| **DCR C** | 8011 | 0D |
| **JNZ Back** | 8012,8013,8014 | C2,09,80 |
| **MOV A, B** | 8015 | 78 |
| **STA 8600H** | 8016,8017,8018 | 32,00,86 |
| **RST 5** | 8019 | EF |

**Table 7.1:** Code Explanation



**Fig. 7.1:** Register Output



**Fig. 7.2:** Debugform



**Fig. 7.3:** Output

**INPUT –**

[8500] – 04, [8501] – 56, [8502] – A9, [8503] – 73, [8504] – 82

**OUTPUT –**

02

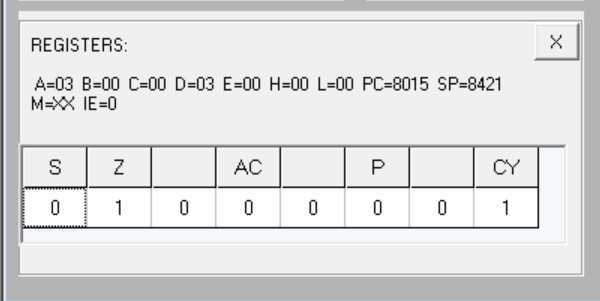
**QUESTION NO: 8**

**OBJECTIVE:**

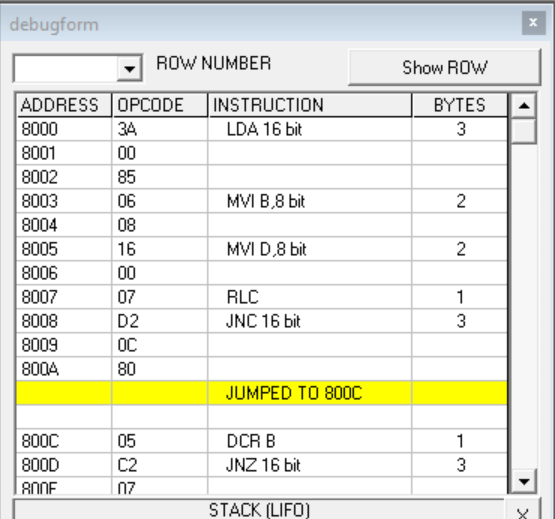
Write a program to count the number of one's in a number.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | 8000,8001,8002 | 3A,00,85 |
| **MVI B, 08** | 8003,8004 | 06,08 |
| **MVI D, 00** | 8005,8006 | 16,00 |
| **Loop1: RLC** | 8007 | 07 |
| **JNC Loop2** | 8008,8009,800A | D2,0C,80 |
| **INR D** | 800B | 14 |
| **Loop2: DCR B** | 800C | 05 |
| **JNZ Loop1** | 800D,800E,800F | C2,07,80 |
| **MOV A, D** | 8010 | 7A |
| **STA 8600H** | 8011,8012,8013 | 32,00,86 |
| **RST 5** | EF | EF |

**Table 8.1:** Code Explanation



**Fig. 8.1:** Register Output



**Fig. 8.2:** Debugform



**Fig. 8.3:** Output

**INPUT –**

[8500] – 25 0010 0101

**OUTPUT –**

[8600] – 03

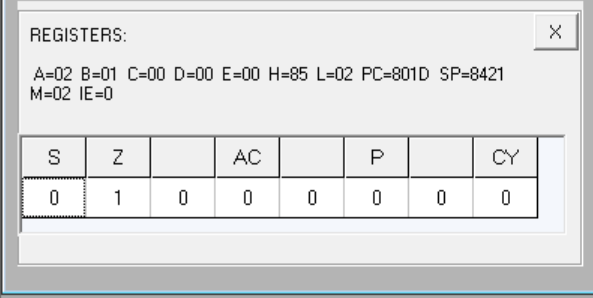
**QUESTION NO: 9**

**OBJECTIVE:**

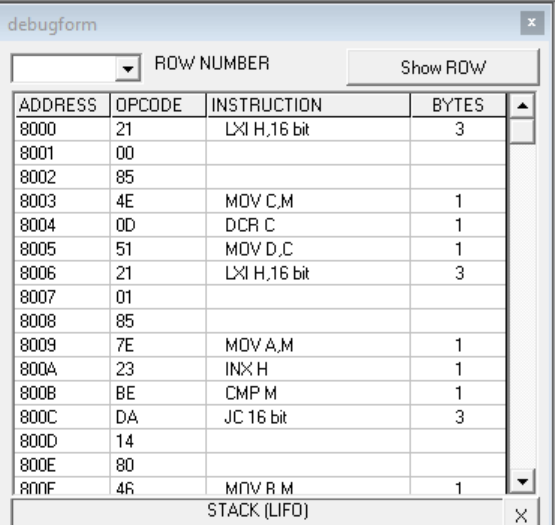
Write a program to arrange numbers in Ascending order.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LXI H, 8500H** | 8000,8001,8002 | 21,00,85 |
| **MOV C, M** | 8003 | 4E |
| **DCR C** | 8004 | 0D |
| **Repeat: MOV D, C** | 8005 | 51 |
| **LXI H, 8501H** | 8006,8007,8008 | 21,01,85 |
| **Loop: MOV A, M** | 8009 | 7E |
| **INX H** | 800A | 23 |
| **CMP M** | 800B | BE |
| **JC Skip** | 800C,800D,800E | DA,14,80 |
| **MOV B, M** | 800F | 46 |
| **MOV M , A** | 8010 | 77 |
| **DCX H** | 8011 | 2B |
| **MOV M, B** | 8012 | 70 |
| **INX H** | 8013 | 23 |
| **Skip: DCR D** | 8014 | 15 |
| **JNZ Loop** | 8015,8016,8017 | C2,09,80 |
| **DCR C** | 8018 | 0D |
| **JNZ Repeat** | 8019,801A,801B | C2,05,80 |
| **RST5** | **801C** | **EF** |

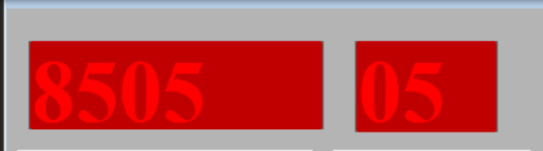
**Table 8.1:** Code Explanation



**Fig. 8.1:** Register Output



**Fig. 8.2:** Debugform



**Fig. 8.3:** Output

**INPUT –**

[8500] – 05, [8501] – 05, [8502] – 04, [8503] – 03, [8504] – 02, [8505] – 01

**OUTPUT –**

[8500] – 05, [8501] – 01, [8502] – 02, [8503] – 03, [8504] – 04, [8505] – 05

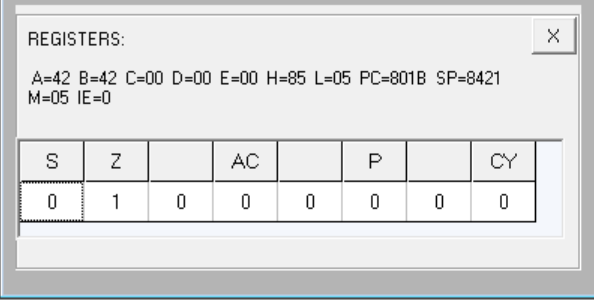
**QUESTION NO: 10**

**OBJECTIVE:**

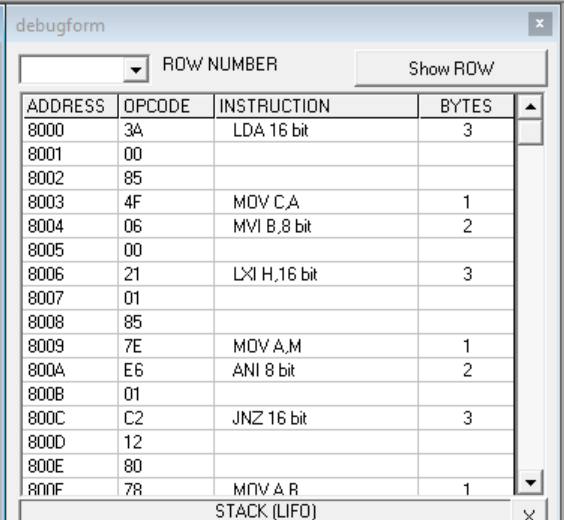
Write a program to calculate the sum of series of even numbers.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | 8000,8001,8002 | 3A,00,85 |
| **MOV C, A** | 8003 | 4F |
| **MVI B, 00** | 8004,8005 | 06 ,00 |
| **LXI H, 8501H** | 8006,8007,8008 | 21,01,85 |
| **Back: MOV A, M** | 8009 | 7E |
| **ANI 01** | 800A,800B | E6,01 |
| **JNZ Skip** | 800C,800D,800E | C2,12,80 |
| **MOV A, B** | 800F | 78 |
| **ADD M** | 8010 | 86 |
| **MOV B, A** | 8011 | 47 |
| **Skip: INX H** | 8012 | 23 |
| **DCR C** | 8013 | 0D |
| **JNZ Back** | 8014,8015,8016 | C2,09,80 |
| **STA 8600H** | 8017,8018,8019 | 32,00,86 |
| **RST 5** | 801A | EF |

**Table 10.1:** Code Explanation



**Fig. 10.1:** Register Output



**Fig. 10.2:** Debugform



**Fig. 10.3:** Output

**INPUT –**

[8500] – 04, [8501] – 20, [8502] – 15 , [8503] – 13, [8504]–22

**OUTPUT –**

[8600] – 42

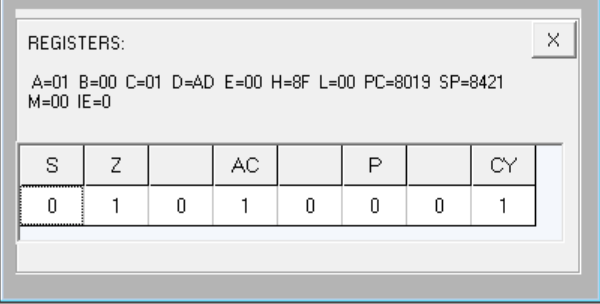
**QUESTION NO: 11**

**OBJECTIVE:**

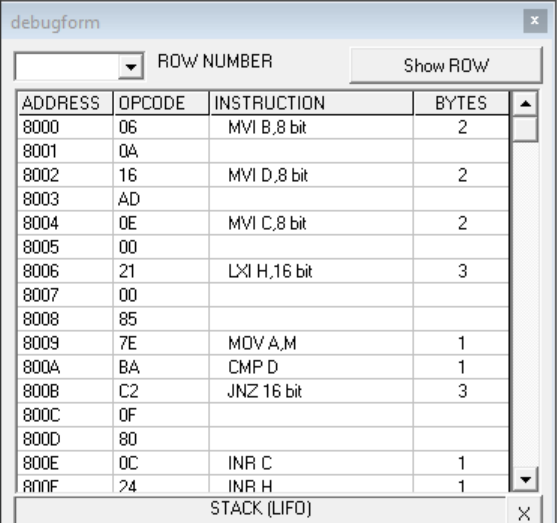
Write an assembly language program to verify how many bytes are present in a given set, which resembles 10101101 in 8085.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **MVI B, 0A** | 8000,8001 | 06,0A |
| **MVI D, AD** | 8002,8003 | 16, AD |
| **MVI C, 00** | 8004,8005 | 0E,00 |
| **LXI H, 8500H** | 8006,8007,8008 | 21,00,85 |
| **Back: MOV A, M** | 8009 | 7E |
| **CMP D** | 800A | BA |
| **JNZ Next** | 800B,800C,800D | C2,0F,80 |
| **INR C** | 800E | 0C |
| **Next: INX H** | 800F | 24 |
| **DCR B** | 8010 | 05 |
| **JNZ Back** | 8011,8012,8013 | C2,09,80 |
| **MOV A, C** | 8014 | 79 |
| **STA 8600H** | 8015,8016,8017 | 32, 00,86 |
| **RST 5** | 8018 | EF |

**Table 11.1:** Code Explanation



**Fig. 11.1:** Register Output



**Fig. 11.2:** Debugform



**Fig. 11.3:** Output

**INPUT –**

[8500] – AD, [8501] – 01, [8502] – 01, [8503] – 01, [8504] – 01, [8505] – 01, [8506] – 01,

[8507] – 01, [8508] – 01, [8509] – 01

**OUTPUT –**

[8600] – 01

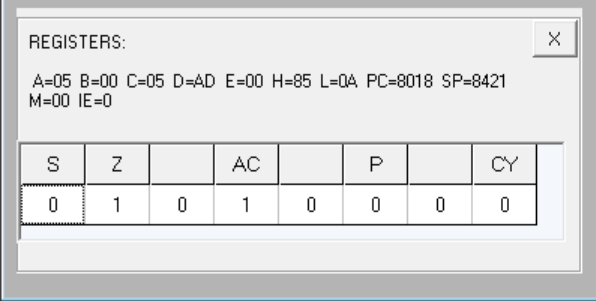
**QUESTION NO: 12**

**OBJECTIVE:**

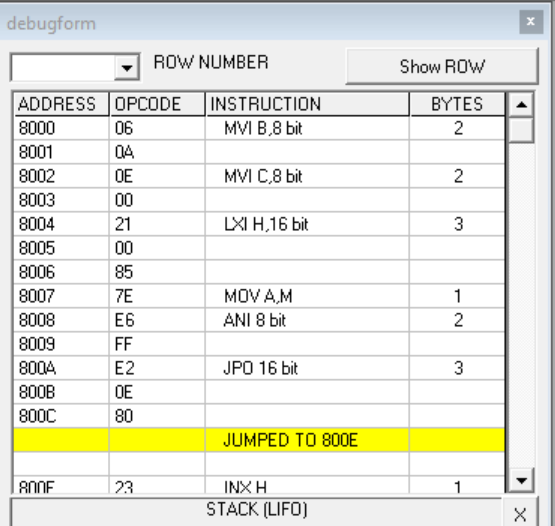
Write an assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **MVI B, 0A** | 8000,8001 | 06,0A |
| **MVI C, 00** | 8002,8003 | 0E,00 |
| **LXI H, 8500H** | 8004,8005,8006 | 21,00,85 |
| **Back: MOV A, M** | 8007 | 7E |
| **ANI FF** | 8008,8009 | E6,FF |
| **JPO Next** | 800A,800B,800C | E2,0E,80 |
| **INR C** | 800D | 0C |
| **Next: INX H** | 800E | 23 |
| **DCR B** | 800F | 05 |
| **JNZ Back** | 8010,8011,8012 | C2,07,80 |
| **MOV A, C** | 8013 | 79 |
| **STA 8600H** | 8014,8015,8016 | 32,00,86 |
| **RST 5** | 8017 | EF |

**Table 12.1:** Code Explanation



**Fig. 12.1:** Register Output



**Fig. 12.2:** Debugform



**Fig. 12`.3:** Output

**INPUT –**

[8500] – 01, [8501] – 03, [8502] – 01, [8503] – 03, [8504] – 01, [8505] – 03, [8506] – 01,

[8507] – 03, [8508] – 01, [8509] – 03

**OUTPUT –**

[8600] – 05

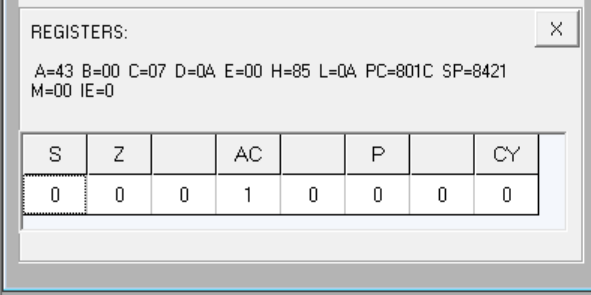
**QUESTION NO: 13**

**OBJECTIVE:**

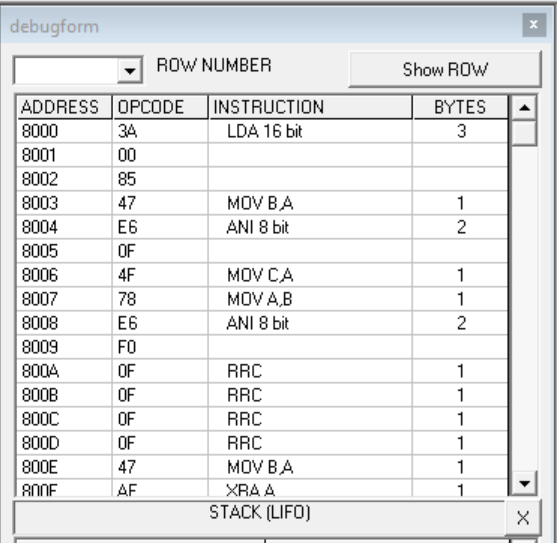
Write an assembly language program to convert a BCD number into its equivalent binary in 8085.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | **8000,8001,8002** | **3A,00,85** |
| **MOV B, A** | **8003** | **47** |
| **ANI 0F** | **8004,8005** | **E6,0F** |
| **MOV C, A** | **8006** | **4F** |
| **MOV A, B** | **8007** | **78** |
| **ANI F0** | **8008,8009** | **E6,F0** |
| **RRC** | **800A** | **0F** |
| **RRC** | **800B** | **0F** |
| **RRC** | **800C** | **0F** |
| **RRC** | **800D** | **0F** |
| **MOV B, A** | **800E** | **47** |
| **XRA A** | **800F** | **AF** |
| **MVI D, 0A** | **8010,8011** | **16,0A** |
| **Sum: ADD D** | **8012** | **82** |
| **DCR B** | **8013** | **05** |
| **JNZ Sum** | **8014,8015,8016** | **C2,12,80** |
| **ADD C** | **8017** | **81** |
| **STA 8600H** | **8018,8019,801A** | **32,00,86** |
| **RST 5** | **801B** | **EF** |

**Table 13.1:** Code Explanation



**Fig. 13.1:** Register Output



**Fig. 13.2:** Debugform



**Fig. 13.3:** Output

**INPUT –**

[8500] – 67

**OUTPUT –**

[8600] – 43

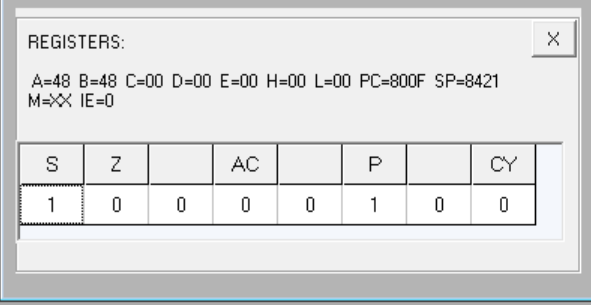
**QUESTION NO: 14**

**OBJECTIVE:**

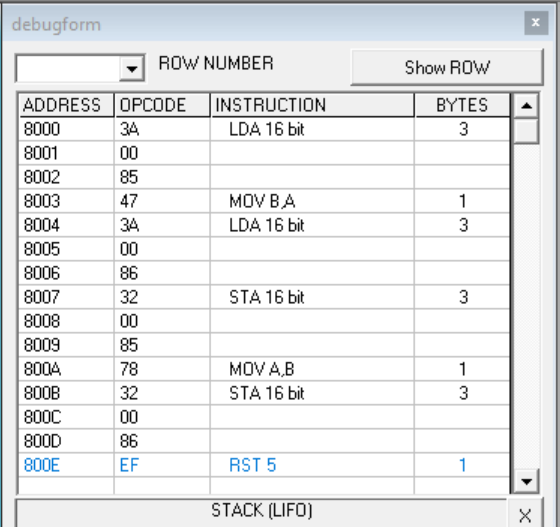
Write an ALP for exchange the contents of memory location.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **LDA 8500H** | 8000, 8001, 8002 | 3A, 00, 85 |
| **MOV B, A** | 8003 | 47 |
| **LDA 8600H** | 8004,8005,8006 | 3A, 00, 86 |
| **STA 8500H** | 8007,8008,8009 | 32, 00, 85 |
| **MOV A, B** | 800A | 78 |
| **STA 8600H** | 800B,800C,800D | 32, 00, 86 |
| **RST 5** | 800E | EF |

**Table 14.1:** Code Explanation



**Fig. 14.1:** Register Output



**Fig. 14.2:** Debugform



**Fig. 14.3:** Output

**INPUT –**

[8500] – 48, [8600] – 88

**OUTPUT –**

[8500] – 88, [8600] – 48

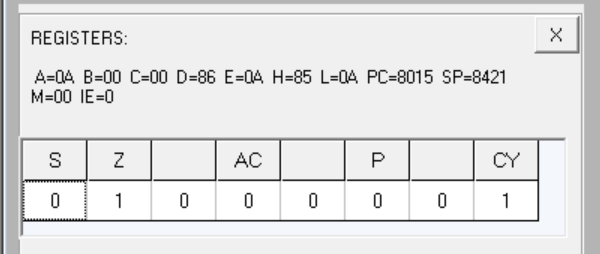
**QUESTION NO: 15**

**OBJECTIVE:**

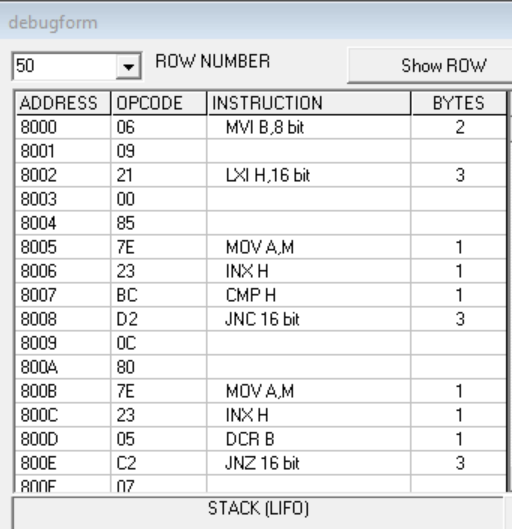
Write a program to find the largest number in an array of 10 elements.

|  |  |  |
| --- | --- | --- |
| **CODE** | **MEMORY LOCATION** | **OPCODE** |
| **MVI B, 09** | 8000,8001 | 06,09 |
| **LXI H, 8500H** | 8002,8003,8004 | 21,00,85 |
| **MOV A, M** | 8005 | 7E |
| **INX H** | 8006 | 23 |
| **Back: CMP M** | 8007 | BC |
| **JNC Next** | 8008,8009,800A | D2,0C,80 |
| **MOV A, M** | 800B | 7E |
| **Next: INX H** | 800C | 23 |
| **DCR B** | 800D | 05 |
| **JNZ Back** | 800E,800F,8010 | C2,07,80 |
| **STA 850AH** | 8011,8012,8013 | 32,0A,85 |
| **RST 5** | 8014 | EF |

**Table 15.1:** Code Explanation



**Fig. 15.1:** Register Output



**Fig. 15.2:** Debugform



**Fig. 15.3:** Output

**INPUT –**

[8500] – 01, [8501] – 02, [8509] – 0A

**OUTPUT –**

[850A] – 0A

**QUESTION NO: 16**

**OBJECTIVE:**

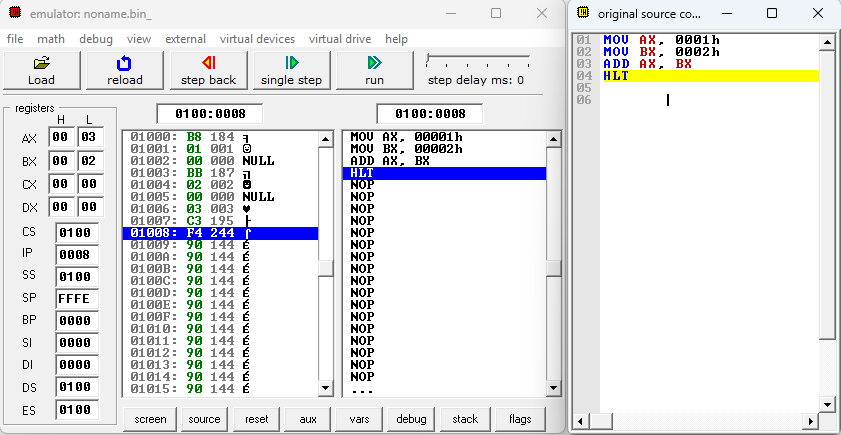
Write an assembly language program to add two 16-bit numbers in 8086.

MOV AX, 0001h

MOV BX, 0002h

ADD AX, BX

HLT



**Fig. 16.1:** Code Compilation and output

**OUTPUT –**

AX = 0003h

BX = 0002h

**QUESTION NO: 17**

**OBJECTIVE:**

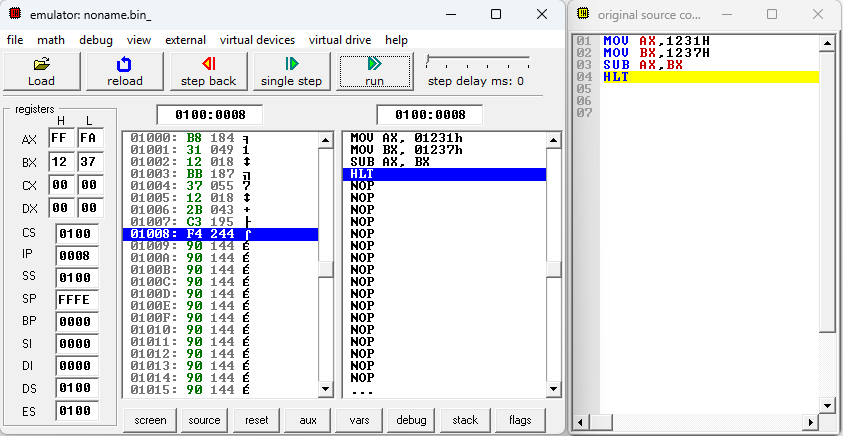
Write an assembly language program to subtract two 16-bit numbers in 8086.

MOV AX,1231H

MOV BX,1237H

SUB AX,BX

HLT



**Fig. 17.1:** Code Compilation and output

**OUTPUT –**

AX = FFFDh

BX = 1237h

**QUESTION NO: 18**

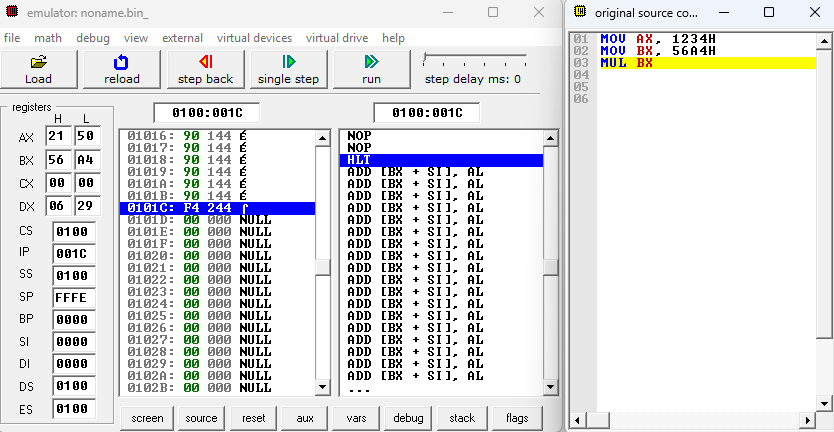
**OBJECTIVE:**

Write an assembly language program to multiply two 16-bit numbers in 8086.

MOV AX, 1234H

MOV BX, 56A4H

MUL BX



**Fig. 18.1:** Code Compilation and output

**OUTPUT –**

AX = 2150h

BX = 0629h

**QUESTION NO: 19**

**OBJECTIVE:**

Write an assembly language program to divide two 16-bit numbers in 8086.

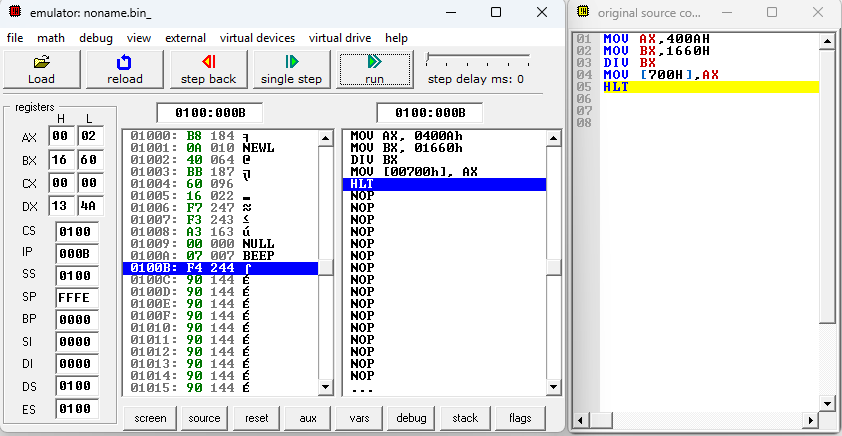
MOV AX, 400AH

MOV BX, 1660H

DIV BX

MOV [700H], AX

HLT



**Fig. 19.1:** Code Compilation and output

**OUTPUT –**

AX = 0002h (quotient)

DX = 134Ah (remainder)

**QUESTION NO: 20**

**OBJECTIVE:**

Write an assembly language program to demonstrate AAA, AAS, AAM, AAD, DAA and DAS in 8086.

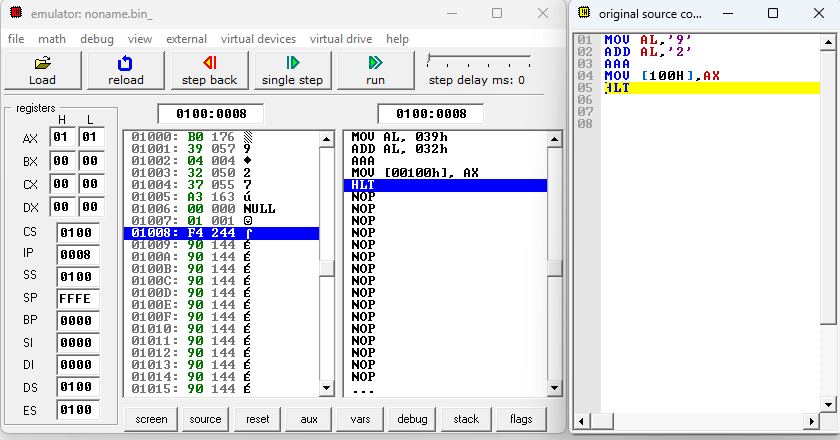
1. **AAA**

MOV AL,'9'

ADD AL,'2'

AAA

MOV [100H],AX

HLT

**Fig. 20.1:** Code Compilation and output

**OUTPUT –**

AX = 0101h

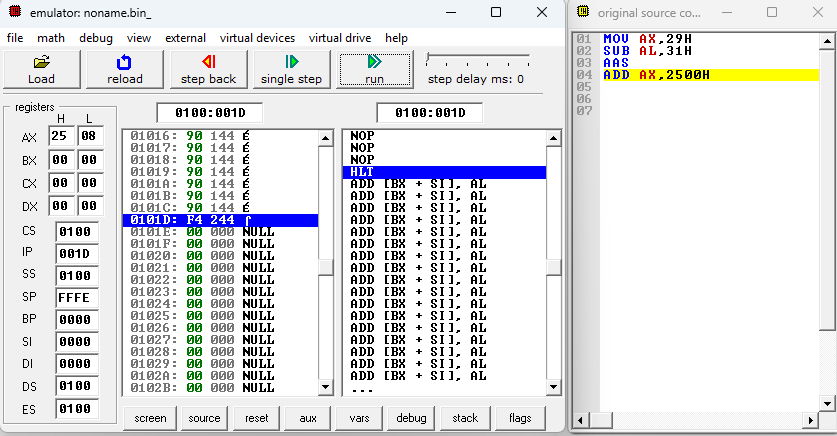
1. **AAS**

MOV AX,29H

SUB AL,31H

AAS

ADD AX,2500H



**Fig. 20.2:** Code Compilation and output

**OUTPUT –**

AX = 2508h

**iii. AAM**

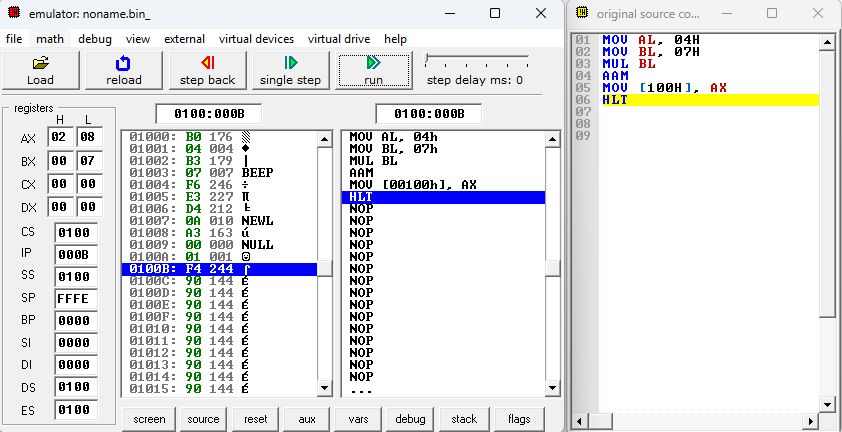
MOV AL, 05H

MOV BL, 09H

## MUL BL

AAM

MOV [100H], AX

 HLT

**Fig. 20.3:** Code Compilation and output

**OUTPUT –**

AX = 2508h

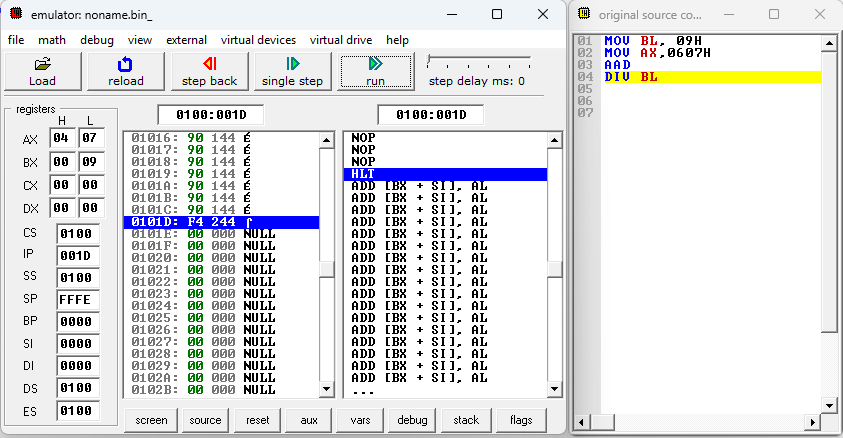
**iv. AAD**

MOV BL, 09H

MOV AX,0607H

AAD

DIV BL



**Fig. 20.4:** Code Compilation and output

**OUTPUT –**

AX = 0407h

**v. DAA**

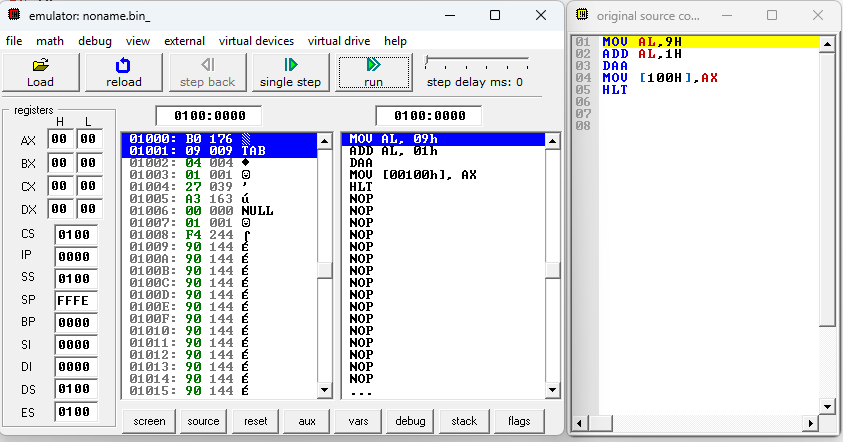
MOV AL,9H

ADD AL,1H

DAA

MOV [100H],AX

HLT



**Fig. 20.5:** Code Compilation and output

**vi. DAS**

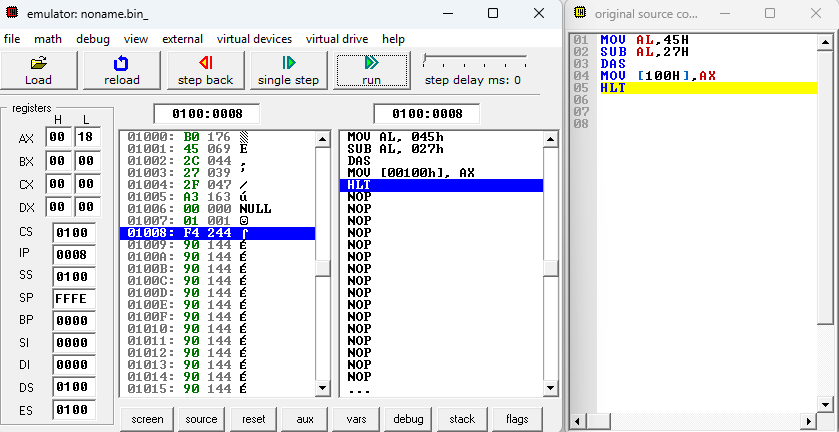
MOV AL,45H

SUB AL,27H

DAS

MOV [100H],AX

HLT



**Fig. 20.6:** Code Compilation and output

**OUTPUT –**

AX = 0018h

**QUESTION NO: 21**

**OBJECTIVE:**

Write an assembly language program to find out the count of positive numbers and negative numbers from a series of signed numbers in 8086.

MOV CL, 0AH

MOV BL, 00H

MOV DL, 00H

LEA SI, [1000H]

L1: MOV AL, [SI]

SHL AL, 01

JNC L2

INC DL

JMP L3

L2: INC BL

L3: INC SI

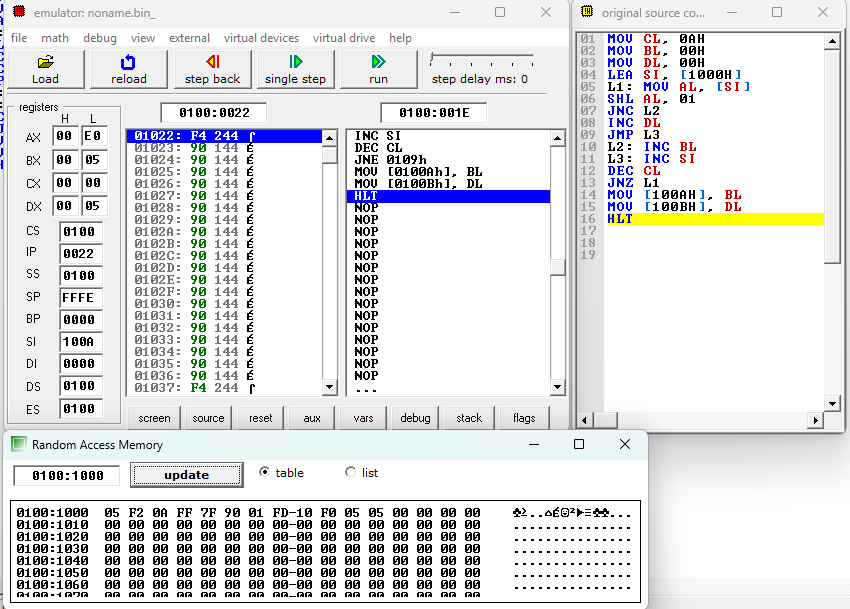
DEC CL

JNZ L1

MOV [100AH], BL

MOV [100BH], DL

HLT



**Fig. 21.1:** Code Compilation and output

**INPUT –**05, F2, 0A, FF, 7F, 90, 01, FD, 10, F0

**OUTPUT – [**100AH]=5 (positive) [100BH]=5 (negative)

**QUESTION NO: 22**

**OBJECTIVE:**

Write an assembly language program to convert to find out the largest number from a given unordered array of 8-bit numbers, stored in the locations starting from a known address in 8086.

MOV CL, 0AH

LEA SI, [1000H]

MOV AL, [SI]

L1: INC SI

MOV BL, [SI]

CMP AL, BL

JC L2

JMP L3

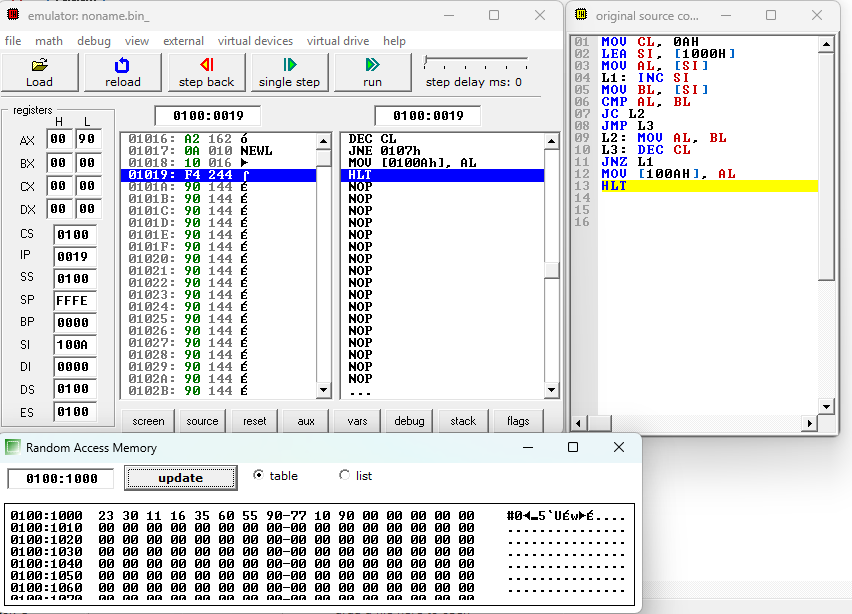
L2: MOV AL, BL

L3: DEC CL

JNZ L1

MOV [100AH], AL

HLT



**Fig. 22.1:** Code Compilation and output

**INPUT-** 23, 30, 11, 16, 35, 60, 55, 90, 77, 10

**OUTPUT-** [100AH]= 90

**QUESTION NO: 23**

**OBJECTIVE:**

Write an assembly language program to convert to find out the largest number from a given unordered array of 16-bit numbers, stored in the locations starting from a known address in 8086.

LEA SI, [1200H]

MOV CL,[SI]

INC SI

MOV AX,[SI]

DEC CL

L2: INC SI

CMP AX,[SI]

JNB L1

MOV AL,[SI]

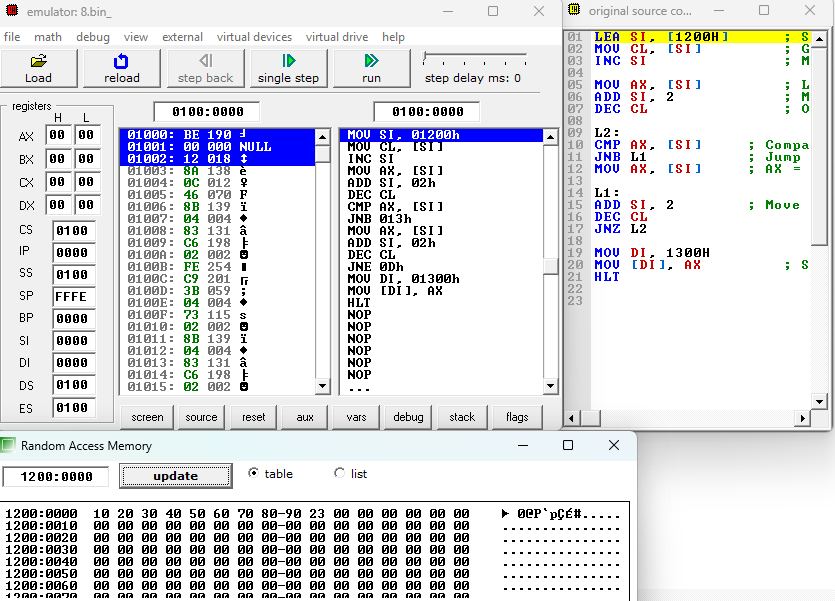
L1: DEC CL

JNZ L2

MOV DI,1300H

MOV [DI] , AX

HLT



**Fig. 23.1:** Code Compilation and output

**INPUT-** 10, 20, 30, 40, 50, 60, 70, 80, 90, 23

**OUTPUT-** 90

**QUESTION NO: 24**

**OBJECTIVE:**

Write an assembly language program to print Fibonacci series in 8086.

LEA SI,[100H]

MOV CL,07H

MOV AX,00H

MOV BX,01H

L1:ADD AX, BX

MOV [SI],AX

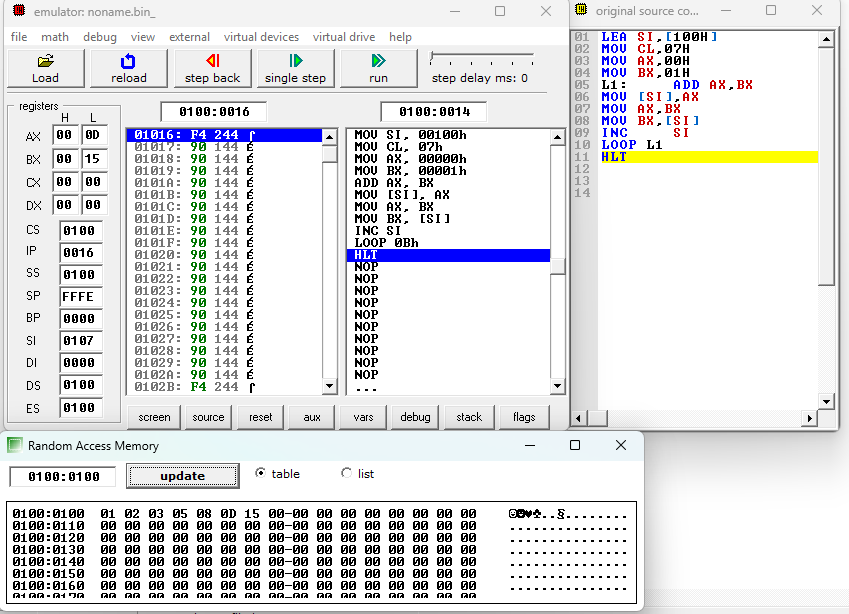
MOV AX,BX

MOV BX,[SI]

INC SI

LOOP L1

HLT



**Fig. 24.1:** Code Compilation and output

**OUTPUT-**

[100H] = 00 (Fibonacci number 0), [101H] = 01 (Fibonacci number 1)

[102H] = 01 (Fibonacci number 1), [103H] = 02 (Fibonacci number 2)

[104H] = 03 (Fibonacci number 3), [105H] = 05 (Fibonacci number 5)

[106H] = 08 (Fibonacci number 8), [107H] = 13 (Fibonacci number 13)

**QUESTION NO: 25**

**OBJECTIVE:**

Write an assembly language program to perform the division 15/6 using the ASCII codes. Store the ASCII codes of the result in register DX.

MOV AX,'15'

ADD BX,'6'

SUB AX,3030H

SUB BL,30H

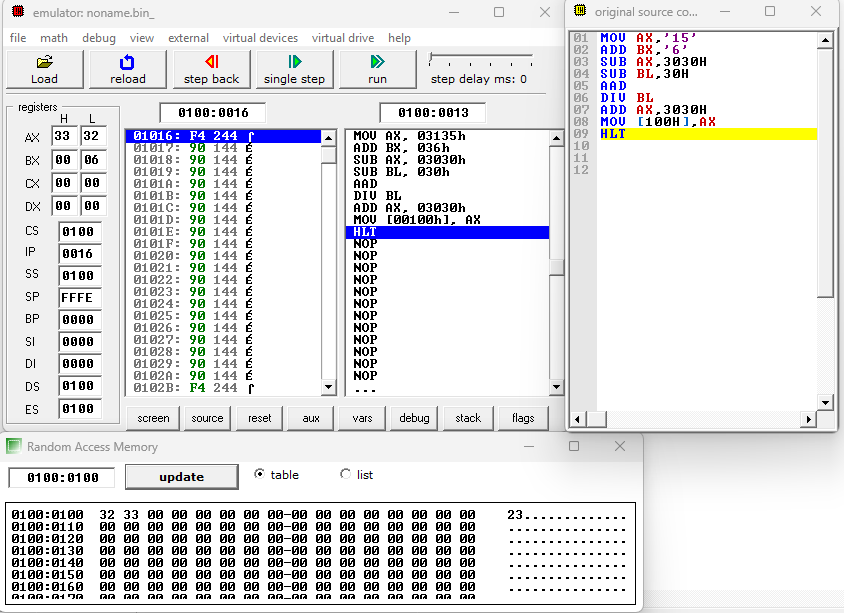
AAD

DIV BL

ADD AX,3030H

MOV [100H],AX

HLT



**Fig. 25.1:** Code Compilation and output

**OUTPUT -** AX= 3332