**MICROPROCESSOR AND ASSEMBLY LANGUAGE PROGRAMMING**

**(3341101)**

**DIPLOMA 4TH SEMESTER**

|  |
| --- |
|  |

**LABORATORY MANUAL**

|  |
| --- |
|  |

**Compiled by**

**Milav J Dabgar**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**GOVERNMENT POLYTECHNIC, PALANPUR.**

**GOVT. POLYTECHNIC, PALANPUR**

**Electronics & Communication Department**

**Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Title** | **Date** | **Grade** | **Sign** |
| 1. | Introduction to 8085 Simulator GNUSim8085. |  |  |  |
| 2. | To study Micro Processor 8085 trainer kit (DYNA) |  |  |  |
| 3. | Write a program to add two 8 bit numbers. |  |  |  |
| 4 | Write a program to subtract two 8 bit numbers. |  |  |  |
| 5 | Write a program to add two 16 bit numbers. |  |  |  |
| 6. | Write a program to subtract two 16 bit numbers. |  |  |  |
| 7. | Write a program to multiply two 8-bit numbers. |  |  |  |
| 8 | Write a program to divide two 8-bit numbers. |  |  |  |
| 9. | Write a program to find 1’s and 2’s Complement of a number |  |  |  |
| 10. | Write a program to find maximum number from block of data. |  |  |  |
| 11 | Write a program to find minimum number from block of data. |  |  |  |
| 12 | Write a program to transfer a block of ten bytes from source to destination memory location. |  |  |  |
| 13 | Write a Program to sort the number in ascending order. |  |  |  |
| 14 | Write a Program to sort the number in descending order. |  |  |  |
| 15 | Write a program to find sum of Block of Numbers stores in memory. |  |  |  |

Practical-1

# Aim:

Introduction to 8085 Simulator GNUSim8085.

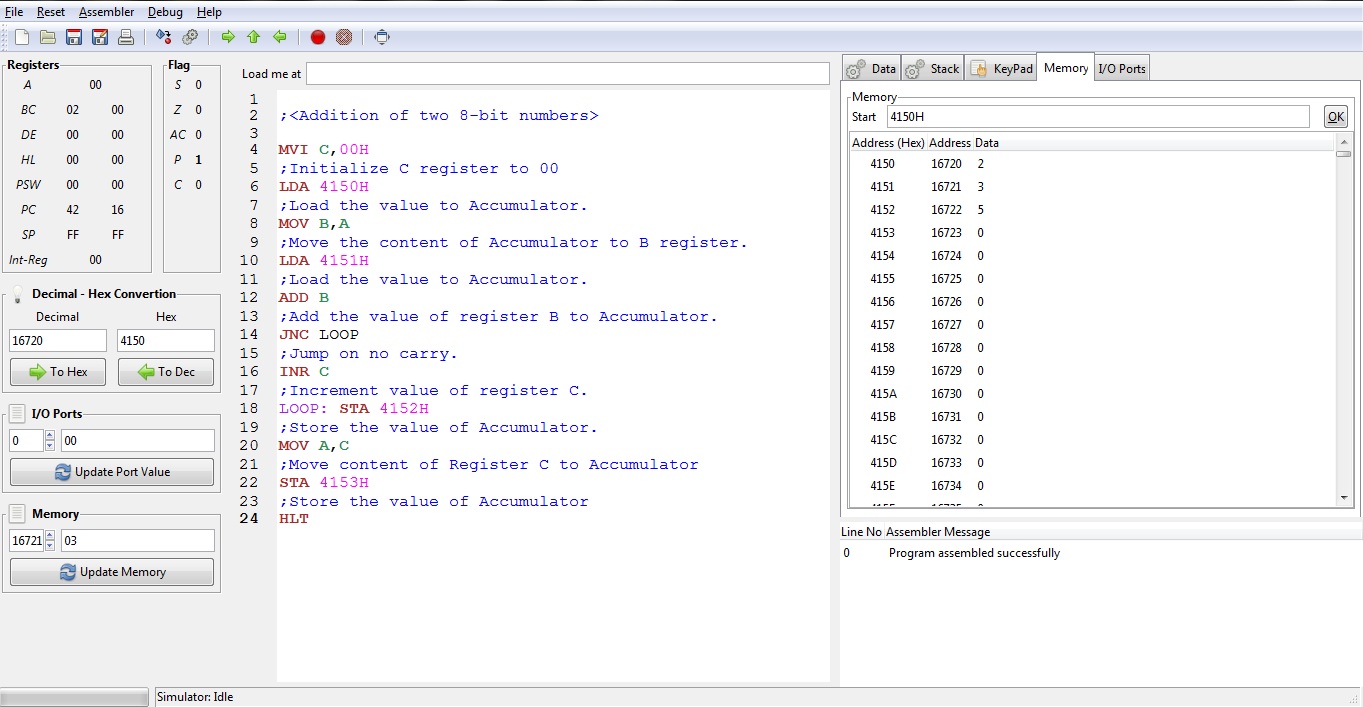
# Features:

GNUSim8085 is a graphical simulator, assembler and debugger for the Intel 8085 microprocessor in Linux and Windows. Some of the features of GNUSim8085 are:

* A simple editor component with syntax highlighting.
* A keypad to input assembly language instructions with appropriate arguments.
* Easy view of register contents
* Easy view of flag contents.
* Hexadecimal <–> Decimal converter.
* View of stack, memory and I/O contents.
* Support for breakpoints for program debugging.
* Stepwise program execution.
* One click conversion of assembly program to Opcode listing.
* Printing support.

# Introduction:

Now we will go through some important sections of the tool. The below is screenshot of the GNUSim8085 Workspace.



It contains below listed main sections:

1. **Editor:** The editor is a part of workspace that lets user enter the code in assembly language. After editing the code user need to save the file in ‘.asm’ format.
2. **Registers & Flags Monitor:** This pane contains all the register of 8085 A, BC, DE, HL, PSW, PC, SP, and Flags: S, Z, AC, P and C. We can monitor the status of each of them during debugging as and when they are updated.
3. **Decimal to Hexadecimal and vice versa convertor:** We can covert a number from Hexadecimal to Decimal and vice versa using tool provided in GNUSim8085. For doing this we need to enter our Hex number (e.g. 4150H) in Box below HEX and as we press the button ‘To Dec’ we get the number in decimal form in the box below ‘Decimal’. The identical process to be followed for converting a number from decimal to hexadecimal number.
4. **I/O Updater:** We can update data on any I/O port by providing its address and valid desired data by pressing ‘Update Port Value’ button.
5. **Memory Updater:** We can update data on any memory address by providing its address and valid desired data by pressing ‘Update Memory’
6. **Data Monitor:** This pane display all the local and global variables declared and defined if any.
7. **Stack Monitor:** It shows the content of the Stack in real time.
8. **Keypad:** This Part of workspace contains all the assembly language instructions. We can select the instruction that we want to use into editor, the tool also guides with proper operand to the instruction.
9. **Memory Monitor:** This sections shows the whole memory of the 8085. We can go to any memory location by entering the address of that particular location. We can enter the address either in Hexadecimal or in decimal form.
10. **I/O Monitor:** This sections shows all I/O Ports of the 8085. We can go to any I/O port by entering the address of that particular location. We can enter the address either in Hexadecimal or in decimal form.
11. **Assembler and Debugger:** GNUSim8085 converts assembly language code into machine language code. We can assemble our code by pressing shortcut key ‘F8’. For Executing the program at once press key ‘F9’. We can also debug the program by executing program one line at a time by pressing key ‘F5’. GNUSim8085 also provide functionality of using breakpoints. Finally we can get the machine codes by the option ‘Show Listing’, shortcut key for the same is ‘CTRL+L’

Practical-2

# Aim:

To study Micro Processor 8085 trainer kit (DYNA)

# Theory:

The Dyna-85 is a low-cost, easy-to-use, entry level, Microprocessor trainer kit. The Dyna-85 kit has Intel's 8085 microprocessor @ 3 MHz together with 8255 as well as 8155 chips onboard. This allows the user to learn all the required peripheral chips together with its 4Kb powerful firmware, it is one of the best products to learn microprocessors especially the 8085 chip. The Dyna-85 allows you to expand your system using its powerful buffered 50 pin expansion bus by building/adding study cards. The Dyna-85 also can be connected to a host PC through the serial port.

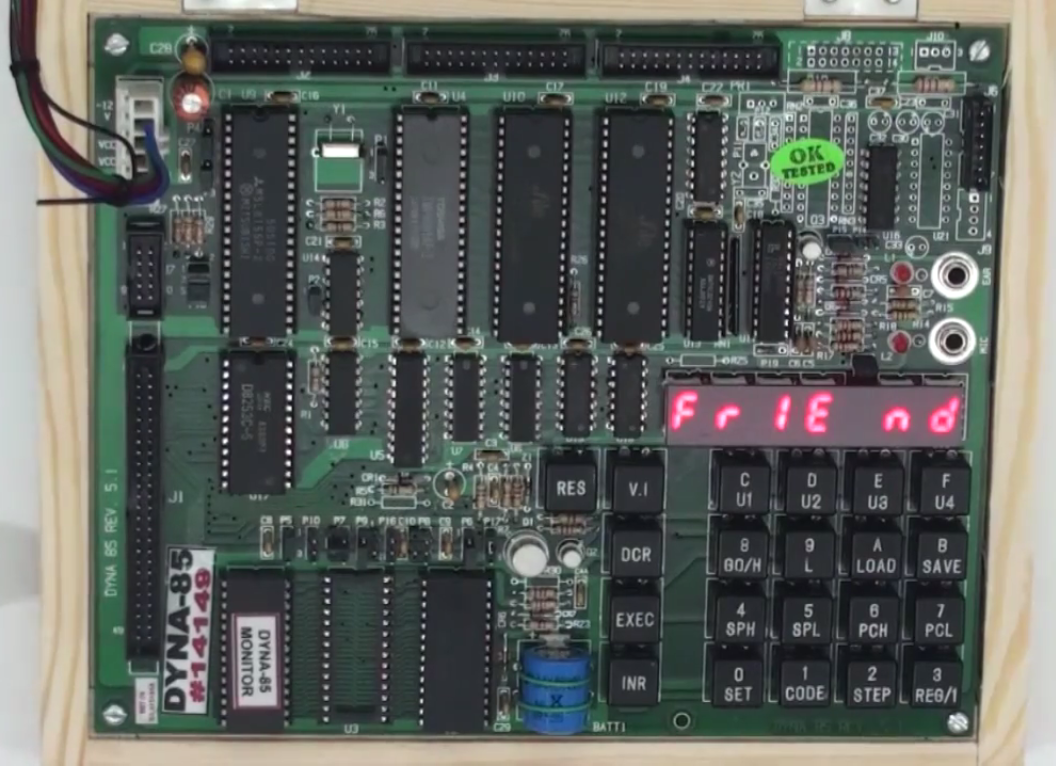


Figure 1Dyna-85 Kit

# Features:

* High performance 8085A CPU @ 3 MHz
* 4 K powerful monitor FIRMWARE. Including all standard commands, codes, functions and utility subroutines.
* 2 K user RAM 6116.
* Three 28 pin sockets provided for memory expansion up to a maximum of 56 K.
* Versatile Keyboard/Display controller using 8279.
* 46 parallel I/O lines, 22 from 8155 and 24 from 8255.
* Serial I/O through auto adjusting type RS -232 channel.
* Built-in audio cassette interface.
* Programmable timer.
* Powerful 8085 interrupt capabilities.
* 6 digit seven segment LED display.
* Highly reliable multi-function keypad.
* All address, data and control and hardware interrupt lines are brought out on a 50 pin FRC connector for system interfacing and expansion.
* Three 16 bit Timer/Counter channels are available on-board, using 8253.
* These channels are available on a 10 pin FRC connector.
* RAM sockets are provided with battery backup.
* Supplied in attractive polished wooden enclosure.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| C000 | 3E 02 |  | MVI A,02H | Get 02H in Accumulator |
| C002 | 06 03 |  | MVI B,03H | Get 03H in Register B |
| C004 | 80 |  | ADD B | Add Content of Register B with that of A |
| C007 | 32 50 C0 |  | STA C050 | Store Result from Accumulator to Memory Location C050H |
| C008 | CF |  | RST 1 | Stop Execution |

# Procedure:

* Connect Power Supply to Dyna-85 Kit and Switch it ON.
* Check that FRIEND is displayed on the SSD LED Display in Dyna-85 Kit, It indicates that the kit is in working condition.
* For Loading the program, Reset the kit using key ‘RES’.
* Press SET key followed by Starting Address of the Code (e.g. C000H) and press INR Key.
* Enter 1st Machine Code (e.g. 3EH) and again press INR key to enter the next machine code.
* Enter rest of the machine codes in the same manner by pressing INR key each time.
* Press RES after code is loaded into 8085 memory.
* For executing the program, Press GO key followed by Starting Address of the Code (e.g. C000H) and then press EXEC key. FRIEND message is displayed if the program is executed successfully.
* For Checking Results,
  + For Checking contents of specific registers, Press REG key followed by register name (e.g. A) of which you want to check content.
  + For Checking Contents of Specific memory location, Press RES key, Press SET Key, followed by Address of Memory location, Press INR Key in the end. The contents are displayed on LED Display.

# Conclusion:

Practical-3

# Aim:

Write a program to add two 8 bit numbers.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 0E 00 |  | MVI C,00H | Initialize C register to 00. |
| 4202 | 3A 50 41 |  | LDA 4150H | Load the value to Accumulator. |
| 4205 | 47 |  | MOV B,A | Move the content of Accumulator to B register. |
| 4206 | 3A 51 41 |  | LDA 4151H | Load the value to Accumulator. |
| 4209 | 80 |  | ADD B | Add the value of register B to Accumulator. |
| 420A | D2 0E 050 |  | JNC LOOP | Jump on no carry. |
| 420D | 0C |  | INR C | Increment value of register C. |
| 420E | 32 52 41 | LOOP: | STA 4152H | Store the value of Accumulator. |
| 4211 | 79 |  | MOV A,C | Move content of Register C to Accumulator. |
| 4212 | 32 53 41 |  | STA 4153H | Store the value of Accumulator. |
| 4215 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs addition of two 8 bit numbers stored at memory locations: 4150H and 4151H, and stores the result of addition at 4152H.
* Load 1st 8 bit Number at memory location 4150H and 2nd 8 bit Number at memory location 4151H.
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location 4152H.

# Observations:

* Before Execution:
  + 4150H: 34H
  + 4151H:12H
  + 4152H: XX (Don’t Care)
* After Execution:
  + 4150H: 34H
  + 4151H: 12H
  + 4152H: 46H

# Conclusion:

Practical-4

# Aim:

Write a program to subtract two 8 bit numbers.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 0E 00 |  | MVI C,00H | Initialize C register to 00. |
| 4202 | 3A 50 41 |  | LDA 4150H | Load the value to Accumulator. |
| 4205 | 47 |  | MOV B,A | Move the content of Accumulator to B register. |
| 4206 | 3A 51 41 |  | LDA 4151H | Load the value to Accumulator. |
| 4209 | 90 |  | SUB B | Subtract the value of register B from Accumulator. |
| 420A | D2 0E 050 |  | JNC LOOP | Jump on no carry. |
| 420D | 2F |  | CMA | Complement Accumulator Content |
| 420E | 3C |  | INR A | Increment value of register A. |
| 420F | 0C |  | INR C | Increment value of register C. |
| 4210 | 32 52 41 | LOOP: | STA 4152H | Store the value of Accumulator. |
| 4213 | 79 |  | MOV A,C | Move content of Register C to Accumulator. |
| 4214 | 32 53 41 |  | STA 4153H | Store the value of Accumulator. |
| 4217 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs subtraction of two 8 bit numbers stored at memory locations: 4150H and 4151H, and stores the result of addition at 4152H.
* Load 1st 8 bit Number at memory location 4150H and 2nd 8 bit Number at memory location 4151H.
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location 4152H.

# Observations:

* Before Execution:
  + 4150H: 34H
  + 4151H: 12H
  + 4152H: XX (Don’t Care)
* After Execution:
  + 4150H: 34H
  + 4151H: 12H
  + 4152H: 22H

# Conclusion:

Practical-5

# Aim:

Write a program to add two 16 bit numbers.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 2A 50 41 |  | LHLD 4150H | Get first 16-bit number |
| 4203 | EB |  | XCHG | Save first 16-bit number in DE |
| 4204 | 2A 52 41 |  | LHLD 4152H | Get second 16-bit number in HL |
| 4207 | 19 |  | DAD D | Add DE and HE |
| 4208 | 22 54 41 |  | SHLD 4154H | Store 16-bit result in memory locations 4154H and 4155H. |
| 420B | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs addition of two 16 bit numbers stored at memory locations: 4150H (Higher Byte), 4151H (Lower Byte) and 4152H (Higher Byte), 4153H (Lower Byte) and stores the result of addition at 4154H (Higher Byte), 4155H (Lower Byte).
* As the numbers are of 16 bit size and 8085 memory width is 8 bit only, each 16 bit number will occupy two memory cells in the memory: one for Higher Byte and another one for Lower Byte.
* Load 1st 16 bit Number at memory location 4150H (Higher Byte) & 4151H (Lower Byte) and 2nd 16 bit Number at memory location 4152H (Higher Byte) & 4153H (Lower Byte).
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location 4154H (Higher Byte), 4155H (Lower Byte).

# Observations:

* Before Execution:
  + 4150H: 87H
  + 4151H: 65H
  + 4152H: 43H
  + 4153H: 21H
  + 4154H: XX (Don’t Care)
  + 4155H: XX (Don’t Care)
* After Execution:
  + 4150H: 87H
  + 4151H: 65H
  + 4152H: 43H
  + 4153H: 21H
  + 4154H: 44H
  + 4155H: 44H

# Conclusion:

Practical-6

# Aim:

Write a program to add subtract 16 bit numbers.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 21 65 87 |  | LXI H,8765H | Load HL with 9876H data |
| 4203 | 01 21 43 |  | LXI B,4321H | Load BC with 4321H data |
| 4206 | 7D |  | MOV A,L | Move the content of L reg. into Accumulator |
| 4207 | 91 |  | SUB C | Subtract the content of L reg. from Accumulator |
| 4208 | 32 50 41 |  | STA 4150H | Store the Result LSB into 4150H |
| 420B | 7C |  | MOV A,H | Move the content of H reg. into Accumulator |
| 420C | 90 |  | SUB B | Subtract the content of H reg. from Accumulator |
| 420D | 32 51 41 |  | STA 4151H | Store the Result LSB into 4151H |
| 4210 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs subtraction of two 16 bit numbers: 8765H & 4321H.
* The numbers are provided directly in instruction as immediate operands.
* We perform subtraction of one byte at a time and store the result in memory.
* As the numbers are of 16 bit size and 8085 memory width is 8 bit only, each 16 bit number will occupy two memory cells in the memory: one for Higher Byte and another one for Lower Byte.
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location 4151H (Higher Byte), 4150H (Lower Byte).

# Observations:

* Before Execution:
  + 4150H: XX (Don’t Care)
  + 4151H: XX (Don’t Care)
* After Execution:
  + 4150H: 44H
  + 4151H: 44H

# Conclusion:

Practical-7

# Aim:

Write a program to multiply two 8-bit numbers.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 16 00 |  | MVI D,00H | Initialize D register to 00 |
| 4202 | 3E 00 |  | MVI A,00H | Initialize A register to 00 |
| 4204 | 21 50 41 |  | LXI H,4150H | Get Address of the 1st 16 bit number in HL Register pair |
| 4207 | 46 |  | MOV B,M | Get the second number in B register |
| 4208 | 23 |  | INX H | Point to the next memory location |
| 4209 | 4E |  | MOV C,M | Get the second number in C register |
| 420A | 80 | LOOP: | ADD B | Add content of A to reg. B |
| 420B | D2 0F 042 |  | JNC NEXT | Jump on no carry to NEXT |
| 420E | 0C |  | INR C | Increment the content of reg. C |
| 420F | 0D | NEXT: | DCR C | decrement the content of reg. C |
| 4210 | C2 0A 042 |  | JNZ LOOP | Jump on no zero to label LOOP |
| 4213 | 32 52 41 |  | STA 4152H | Store the LSB of result in memory |
| 4216 | 7A |  | MOV A,D | Get the MSB of result in Accumulator |
| 4217 | 32 53 41 |  | STA 4153H | Store the MSB of result in memory |
| 420B | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs multiplication of Two 8 bit numbers stored at memory location 4150H and 4151H.
* We use the method called as repetitive addition. In this method we keep on adding No. 1 (e.g. 2) number of times that is equal to No. 2 (e.g. 3). The result would be multiplication of two 8 bit numbers (3\*4 = 6).
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location 4153H (Higher Byte), 4152H (Lower Byte).

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 02H   + 4151H: 03H   + 4152H: XX (Don’t Care)   + 4153H: XX (Don’t Care) | * After Execution:   + 4150H: 02H   + 4151H: 03H   + 4152H: 06H   + 4153H: 00H |

# Conclusion:

Practical-8

# Aim:

Write a program to divide two 8-bit numbers.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 21 50 41 |  | LXI H,4150H | Get Address of Divisor(4150H) in HL Register Pair |
| 4203 | 46 |  | MOV B,M | Get Divisor into Reg B. |
| 4204 | 0E 00 |  | MVI C,00H | Clear C reg. for quotient. |
| 4206 | 23 |  | INX H | Point to next location (Address of Dividend - 4151H) |
| 4207 | 7E |  | MOV A,M | Get the Dividend into Accumulator |
| 4208 | B8 | NEXT: | CMP B | Compare A Reg with Reg B and update flags |
| 4209 | DA 11 42 |  | JC LOOP | If A<B it will result in carry,  It indicates Division is complete.  Go to label LOOP and save result. |
| 420C | 90 |  | SUB B | Perform Subtraction (Dividend in A – Divisor in B) |
| 420D | 0C |  | INR C | increment the content of reg. C used for Quotient |
| 420E | C3 08 042 |  | JMP NEXT | Perform Repetitive Subtraction (Dividend - Divisor) till Dividend<Divisor. |
| 4211 | 32 52 41 | LOOP: | STA 4152H | Store the Quotient of result in to memory location 4152H |
| 4214 | 79 |  | MOV A,C | Get the remainder of result in Accumulator |
| 4215 | 32 53 41 |  | STA 4153H | Store remainder in to memory location 4153H |
| 4218 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs division of Two 8 bit numbers stored at memory location 4150H (Divisor Address) and 4151H (Dividend Address).
* We use the method called as repetitive subtraction. We keep on subtracting Divisor (e.g. 2) from Dividend (e.g. 7) till Dividend<Divisor.
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location 4153H (Remainder), 4152H (Quotient).

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 02H   + 4151H: 07H   + 4152H: XX (Don’t Care)   + 4153H: XX (Don’t Care) | * After Execution:   + 4150H: 02H   + 4151H: 07H   + 4152H: 03H   + 4153H: 01H |

Practical-9

# Aim:

Write a program to find 1’s and 2’s Complement of a number

# Program:

***Program 9.1: to find 1’s Complement of a number.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 3A 50 41 |  | LDA 4150H | Get the number |
| 4203 | 2F |  | CMA | Complement number |
| 4204 | 32 51 41 |  | STA 4151H | Store the result |
| 4207 | 76 |  | HLT | Stop Execution. |

***Program 9.2 to find 2’s Complement of a number.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 3A 50 41 |  | LDA 4150H | Get the number |
| 4203 | 2F |  | CMA | Complement number |
| 4204 | C6 01 |  | ADI 01H | Add one in the number |
| 4206 | 32 51 41 |  | STA 4151H | Store the result |
| 4209 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program uses CMA and CMA & ADI instruction to find 1’s and 2’s complement of a number respectively. Store the result at a memory location 4150H and stores the result at 4151H.
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location 4151H.

# Observations:

* Before Execution:
  + 4150H: 0AH (Don’t Care)
  + 4151H: XX (Don’t Care)
* After Execution of Program 9.1:
  + 4150H: 0AH
  + 4151H: F5H
* After Execution of Program 9.2:
  + 4150H: 0AH
  + 4151H: F6H

# Conclusion:

Practical-10

# Aim:

Write a program to find maximum number from block of data.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 21 50 41 |  | LXI H,4150H | Load H-L pair with address 4150H |
| 4203 | 4E |  | MOV C,M | Move counter from memory to reg. C. |
| 4204 | 23 |  | INX H | Increment H-L pair. |
| 4205 | 7E |  | MOV A,M | Move the 1’st number from memory to reg. A |
| 4206 | 0D |  | DCR C | Decrement counter. |
| 4207 | 23 | LOOP: | INX H | Increment H-L pair. |
| 4208 | 46 |  | MOV B,M | Move the next number from memory to reg. B |
| 4209 | B8 |  | CMP B | Compare B with A. |
| 420A | D2 0E 042 |  | JNC AHEAD | Jump to AHEAD if there is not a carry. |
| 420D | 78 |  | MOV A,B | Move largest from reg. B to reg. A. |
| 420E | 0D | AHEAD: | DCR C | Decrement counter. |
| 420F | C2 07 042 |  | JNZ LOOP | Jump to LOOP if counter is not zero. |
| 4212 | 23 |  | INX H | Increment H-L pair. |
| 4213 | 77 |  | MOV M,A | Move the result from reg. A to memory. |
| 4214 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program finds largest number from a Block of number. Length of Block (04) is to be provided in the memory location 4150H and Numbers to be provided from memory location 4151H onwards (4551H: 9, 4152H: 4, 4153H: 2, and 4154H: 6).
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location next to last element of Block (e.g. 4155H: 9).

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 04H   + 4153H: 02H   + 4154H: 06H   + 4155H: XX (Don’t Care) | * After Execution of Program 9.1:   + 4150H: 04H   + 4151H: 09H   + 4152H: 04H   + 4153H: 02H   + 4154H: 06H   + 4155H: 09H |

# Conclusion:

Practical-11

# Aim:

Write a program to find minimum number from block of data.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 21 50 41 |  | LXI H,4150H | Load H-L pair with address 4150H |
| 4203 | 4E |  | MOV C,M | Move counter from memory to reg. C. |
| 4204 | 23 |  | INX H | Increment H-L pair. |
| 4205 | 7E |  | MOV A,M | Move the 1’st number from memory to reg. A |
| 4206 | 0D |  | DCR C | Decrement counter. |
| 4207 | 23 | LOOP: | INX H | Increment H-L pair. |
| 4208 | 46 |  | MOV B,M | Move the next number from memory to reg. B |
| 4209 | B8 |  | CMP B | Compare B with A. |
| 420A | DA 0E 042 |  | JC AHEAD | Jump to AHEAD if there is a carry. |
| 420D | 78 |  | MOV A,B | Move largest from reg. B to reg. A. |
| 420E | 0D | AHEAD: | DCR C | Decrement counter. |
| 420F | C2 07 042 |  | JNZ LOOP | Jump to LOOP if counter is not zero. |
| 4212 | 23 |  | INX H | Increment H-L pair. |
| 4213 | 77 |  | MOV M,A | Move the result from reg. A to memory. |
| 4214 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program finds smallest number from a Block of number. Length of Block (04) is to be provided in the memory location 4150H and Numbers to be provided from memory location 4151H onwards (4551H: 9, 4152H: 4, 4153H: 2, and 4154H: 6).
* Assemble and execute the program, Debug if Required.
* Check for Result at memory location next to last element of Block (e.g. 4155H: 2).

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 04H   + 4153H: 02H   + 4154H: 06H   + 4155H: XX (Don’t Care) | * After Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 04H   + 4153H: 02H   + 4154H: 06H   + 4155H: 02H |

# Conclusion:

Practical-12

# Aim:

Write a program to transfer a block of ten bytes from source to destination memory location.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 0E 05 |  | MVI C,05H | Initialize counter |
| 4202 | 21 50 41 |  | LXI H,4150H | Initialize source memory pointer |
| 4205 | 11 50 51 |  | LXI D,5150H | Initialize destination memory pointer |
| 4208 | 7E | BACK: | MOV A,M | Get byte from source memory block |
| 4209 | 12 |  | STAX D | Store byte in the destination memory block |
| 420A | 23 |  | INX H | Increment source memory pointer |
| 420B | 13 |  | INX D | Increment destination memory pointer |
| 420C | 0D |  | DCR C | Decrement counter |
| 420D | C2 08 042 |  | JNZ BACK | If counter 0 repeat |
| 4210 | 76 |  | HLT | Stop Execution. |

# Description:

* The above program transfers Block of data stored in memory from one location to another.
* The Block length is to be specified in program and stored in register C (e.g. 05H)
* Save block of data in to memory starting at location 4150H.
* Assemble and execute the program, Debug if Required.
* Check for result in destination block of data at memory location starting at 5150H.

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 01H   + 4151H: 02H   + 4152H: 03H   + 4153H: 04H   + 4154H: 05H   + 5150H: XX (Don’t Care)   + 5151H: XX (Don’t Care)   + 5152H: XX (Don’t Care)   + 5153H: XX (Don’t Care)   + 5154H: XX (Don’t Care) | * After Execution:   + 4150H: 01H   + 4151H: 02H   + 4152H: 03H   + 4153H: 04H   + 4154H: 05H   + 5150H: 01H   + 5151H: 02H   + 5152H: 03H   + 5153H: 04H   + 5154H: 05H |

# Conclusion:

Practical-13

# Aim:

Write a Program to sort the number in ascending order.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 21 50 41 |  | LXI H,4150H | Store the address of Block Length in HL Pair |
| 4203 | 4E |  | MOV C,M | Get Block Length in to Register C and use it as Counter |
| 4204 | 0D |  | DCR C | Decrement Outer Loop Counter by 1 |
| 4205 | 51 | REPEAT: | MOV D,C | Use D as Inner Loop Counter |
| 4206 | 21 51 41 |  | LXI H,4151H | Store address of the 1st element of the Block in HL Pair |
| 4209 | 7E | LOOP: | MOV A,M | Get 1st element of Block in Accumulator for Processing |
| 420A | 23 |  | INX H | Increment HL Pair content to Point to 2nd element of Block |
| 420B | BE |  | CMP M | Compare A (1st Number) with M (2nd Number) and update flags |
| 420C | DA 14 42 |  | JC SKIP | if A<M Got to label SKIP, and skip swapping of data |
| 420F | 46 |  | MOV B,M | Move 2nd Number from M to B |
| 4210 | 77 |  | MOV M,A | Move 1st Number from A to M |
| 4211 | 2B |  | DCX H | Set HL Pointer to 1st Slot Address |
| 4212 | 70 |  | MOV M,B | Put 2nd Number (Currently in Reg B) into 1st Slot Address (M Pointed by HL) |
| 4213 | 23 |  | INX H | Set HL Pointer to 2nd Slot Address |
| 4214 | 15 | SKIP: | DCR D | Decrement Inner Loop Counter by 1 |
| 4215 | C2 09 042 |  | JNZ LOOP | Go to label LOOP till iteration counter becomes 0 |
| 4218 | 0D |  | DCR C | Decrement Outer Loop Counter by 1 |
| 4219 | C2 05 042 |  | JNZ REPEAT | Go to label REPEAT till Outer Loop Counter Becomes 0 |
| 421C | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs Bubble Sorting and arranges the number into ascending order.
* Length of Block (04) is to be provided in the memory location 4150H and Numbers to be provided from memory location 4151H onwards (4151H: 9, 4152H: 4, 4153H: 2, and 4154H: 6).
* Assemble and execute the program, Debug if Required.
* Check for result in destination block of data at memory location starting at 5150H.

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 04H   + 4153H: 02H   + 4154H: 06H | * After Execution:   + 4150H: 04H   + 4151H: 02H   + 4152H: 04H   + 4153H: 06H   + 4154H: 09H |

# Conclusion:

Practical-14

# Aim:

Write a Program to sort the number in descending order.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 21 50 41 |  | LXI H,4150H | Store the address of Block Length in HL Pair |
| 4203 | 4E |  | MOV C,M | Get Block Length in to Register C and use it as Counter |
| 4204 | 0D |  | DCR C | Decrement Outer Loop Counter by 1 |
| 4205 | 51 | REPEAT: | MOV D,C | Use D as Inner Loop Counter |
| 4206 | 21 51 41 |  | LXI H,4151H | Store address of the 1st element of the Block in HL Pair |
| 4209 | 7E | LOOP: | MOV A,M | Get 1st element of Block in Accumulator for Processing |
| 420A | 23 |  | INX H | Increment HL Pair content to Point to 2nd element of Block |
| 420B | BE |  | CMP M | Compare A (1st Number) with M (2nd Number) and update flags |
| 420C | D2 14 42 |  | JNC SKIP | if A>M Got to label SKIP, and skip swapping of data |
| 420F | 46 |  | MOV B,M | Move 2nd Number from M to B |
| 4210 | 77 |  | MOV M,A | Move 1st Number from A to M |
| 4211 | 2B |  | DCX H | Set HL Pointer to 1st Slot Address |
| 4212 | 70 |  | MOV M,B | Put 2nd Number (Currently in Reg B) into 1st Slot Address (M Pointed by HL) |
| 4213 | 23 |  | INX H | Set HL Pointer to 2nd Slot Address |
| 4214 | 15 | SKIP: | DCR D | Decrement Inner Loop Counter by 1 |
| 4215 | C2 09 042 |  | JNZ LOOP | Go to label LOOP till iteration counter becomes 0 |
| 4218 | 0D |  | DCR C | Decrement Outer Loop Counter by 1 |
| 4219 | C2 05 042 |  | JNZ REPEAT | Go to label REPEAT till Outer Loop Counter Becomes 0 |
| 421C | 76 |  | HLT | Stop Execution. |

# Description:

* The above program performs Bubble Sorting and arranges the number into descending order.
* Length of Block (04) is to be provided in the memory location 4150H and Numbers to be provided from memory location 4151H onwards (4151H: 9, 4152H: 4, 4153H: 2, and 4154H: 6).
* Assemble and execute the program, Debug if Required.
* Check for result in destination block of data at memory location starting at 4151H.

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 04H   + 4153H: 02H   + 4154H: 06H | * After Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 06H   + 4153H: 04H   + 4154H: 02H |

# Conclusion:

Practical-15

# Aim:

Write a program to find sum of Block of Numbers stores in memory.

# Program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **Machine Codes** | **Labels** | **Instructions** | **Comments** |
| 4200 | 21 50 41 |  | LXI H,4150H | set up HL as memory pointer |
| 4203 | 0E 00 |  | MVI C,00H | clear C to save sum |
| 4205 | 41 |  | MOV B,C | Clear B to save carry |
| 4206 | 7E | NXTBYTE: | MOV A,M | Transfer current reading to (A) |
| 4207 | FE 00 |  | CPI 00H | Is it last reading |
| 4209 | CA 16 42 |  | JZ OUTPUT | If yes go to output section |
| 420C | 81 |  | ADD C | Add previous sum to accumulator |
| 420D | D2 11 42 |  | JNC SAVE | Skip CY register if there is no carry |
| 4210 | 04 |  | INR B | Update carry register |
| 4211 | 4F | SAVE: | MOV C,A | Save sum |
| 4212 | 23 |  | INX H | point to next reading |
| 4213 | C3 06 042 |  | JMP NXTBYTE | Go back to get next reading |
| 4216 | 21 50 51 | OUTPUT: | LXI H,5150H | Point to 5150H memory location |
| 4219 | 71 |  | MOV M,C | Store low-order byte of sum in 5150H |
| 421A | 23 |  | INX H | Point to 5151H memory location |
| 421B | 70 |  | MOV M,B | Store high-order byte of sum in 5150H |
| 421C | 76 |  | HLT | Stop Execution. |

# Description:

* This program reads the numbers from the memory location 4150H till it finds 00 (end of input) and then sums up and stores the result in 5150H (LOB), 5151H (HOB) respectively.
* Assemble and execute the program, Debug if Required.

# Observations:

|  |  |
| --- | --- |
| * Before Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 05H   + 4153H: 02H   + 4154H: 00H   + 5150H: XX (Don’t Care) | * After Execution:   + 4150H: 04H   + 4151H: 09H   + 4152H: 05H   + 4153H: 02H   + 4154H: 00H   + 5150H: 20H |

# Conclusion: