**Apex** **College**

*Affiliated* *to Pokhara University*

**LAB** **MANUAL**

**ON**



**MICROPROCESSOR**

**BCIS:** **III SEM**

**Prepared** **By:**

Er. Raju Sharan

Lab Instructor

**INTRODUCTION** **TO** **MICROPROCESSOR** **8085**

>**Aim**

To study the microprocessor 8085

> **Architecture** **of** **8085** **Microprocessor** ***a)*** ***General*** ***purpose*** ***registers***

It is an 8 bit register i.e. B, C, D, E, H, and L. The combination of 8 bit register is known as register pair, which can hold 16 bit data. The HL pair is used to act as memory pointer is accessible to program.

***b)*** ***Accumulator***

It is an 8 bit register which hold one of the data to be processed by ALU and stored the result of the operation.

***c)*** ***Program*** ***counter*** ***(PC)***

It is a 16 bit pointer which maintains the address of a byte entered to line stack. ***d)*** ***Stack*** ***pointer*** ***(SP)***

It is a 16 bit special purpose register which is used to hold line memory address for line next instruction to be executed.

***e)*** ***Arithmetic*** ***and*** ***logical*** ***unit***

It carries out arithmetic and logical operation by 8 bit address it uses the accumulator content as input the ALU result is stored back into accumulator.

***f)*** ***Temporary*** ***register***

It is an 8 bit register associated with ALU hold data, entering an operation, used by the microprocessor and not accessible to programs.

***g)*** ***Flags***

Flag register is a group of fire, individual flip flops line content of line flag register will change after execution of arithmetic and logic operation. The line states flags are

i) Carry flag (C) ii) Parity flag (P) iii) Zero flag (Z)

iv) Auxiliary carry flag (AC) v) Sign flag (S)

***h)*** ***Timing*** ***and*** ***control*** ***unit***

Synchronizes the microprocessor operation with the clock and generates control signal from it necessary to communicate between controller and peripherals.

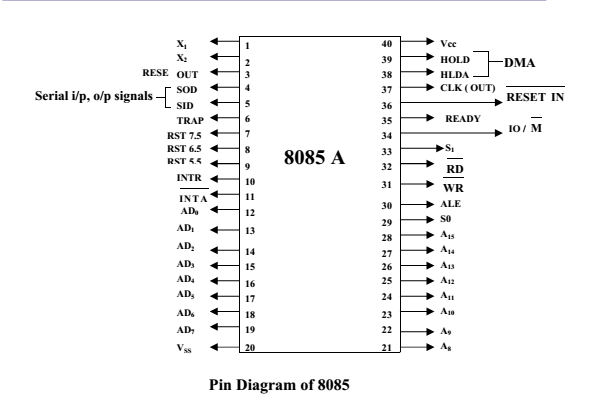
***i)*** ***Instruction*** ***register*** ***and*** ***decoder***

Instruction is fetched from line memory and stored in line instruction register decoder the stored information.

***j)*** ***Register*** ***Array***

These are used to store 8 bit data during execution of some instruction.

**PIN** **Description**



***a) Address*** ***Bus***

1. The pins A0 – A15 denote the address bus. 2. They are used for most significant bit.

***b)*** ***Address*** ***/*** ***Data*** ***Bus***

1. AD0 – AD7 constitutes the address / Data bus. 2. These pins are used for least significant bit.

***c)*** ***ALE:*** ***(Address*** ***Latch*** ***Enable)***

The signal goes high during the first clock cycle and enables the lower order address bits.

***d)*** ***IO*** ***/*** ***M***

1. This distinguishes whether the address is for memory or input. 2. When these pins go high, the address is for an I/O device.

***e)*** ***S0*** ***–*** ***S1***

S0 and S1 are status signal which provides different status and functions.

***f)*** ***RD***

1. This is an active low signal.

2. This signal is used to control READ operation of the microprocessor.

***g)*** ***WR***

1. WR is also an active low signal.

2. Controls the write operation of the microprocessor.

***h)*** ***HOLD***

1. This indicates if any other device is requesting the use of address and data bus.

***i)*** ***HLDA***

1. HLDA is the acknowledgement signal for HOLD.

***j)*** ***INTR***

1. INTE is an interrupt request signal.

2. IT can be enabled or disabled by using software.

**k)** **INTA**

1. Whenever the microprocessor receives interrupt signal 2. It has to be acknowledged.

**l*)*** ***RST*** ***5.5,*** ***6.5,*** ***7.5***

1. These are nothing but the restart interrupts.

2. They insert an internal restart junction automatically.

***m)*** ***TRAP***

1. Trap is the only non-maskable interrupt.

2. It cannot be enabled (or) disabled using program.

***n)*** ***RESET*** ***IN***

1. This pin resets the program counter to 0 to 1 and results interrupt enable and HLDA flip flops.

***o)*** ***X1,*** ***X2***

These are the terminals which are connected to external oscillator to produce the necessary and suitable clock operation.

***p)*** ***SID***

This pin provides serial input data

***q)*** ***SOD***

This pin provides serial output data

***r)*** ***VCC*** ***and*** ***VSS***

**1.** VCC is +5V supply pin

**2.** VSS is ground pin

***s)*** ***Specifications***

**1.** **Processors**

Intel 8085 at E144 MHz clock **2.** **Memory**

Monitor RAM: 0000 – IFFF EPROM Expansion:2000 – 3FFF’s 0000-FFFF

System RAM: 4000 – 5FFF Monitor data area 4100 – 5FFF RAM Expansion 6000 – BFFF

**3.** **Input** **/** **Output**

Parallel: A8 TTL input timer with 2 number of 32-55 only input timer available in

**4.** **Serial:** Only one number RS 232-C, Compatible, crucial interface using 8281A

**5.** **Timer:** 3 channels-16 bit programmable units, using 8253 channel ‘0’ used for no band late.

**6.** **Clock** **generator:** Channel ‘1’ is used for single stopping used program.

**7.** **Display:** 6 digits – 7 segments LED display with filter 4 digit for adder display and 2 digits for data display.

**8.** **Key** **board:** 21 keys, soft keyboard including common keys and hexadecimal keys.

***RES:*** Reset keys allow terminating present activity and retaining to its on initialize state.

***INT:*** Maskable interrupt connect to CPU’s RST 7.5 interrupt.

***DEC:*** Decrement the adder by 1.

***EXEC:*** Execute line particular value after selecting address through go command. ***NEXT:*** Increment the address by 1 and then display its content.

**9.** **System** **Power** **Consumption** Micro BSEB2 MICRO SSEB +5V @ 1Amp +5V@ 800 mA +12V @ 200 mA

- 12V @ 100 mA EE0310-Microprocessor & Microcontroller Lab

**10.** **Power** **Supply** **Specification** MICRO SSEM

230V, AC @ 80 Hz +5V @ 600 mA

> **Enter** **Program** **into** **Trainer** **Kit**

1. Press ‘RESET’ key

2. Sub (key processor represent address field) 3. Enter the address (16 bit) and digit in hex

4. Press ‘NEXT’ key

5. Enter the data

6. Again press “NEXT”

7. Again after taking the program, are use HLT instructions its Hex code 8. Press “NE

> **How** **to** **Execute** **Program** 1. Press “RESET”

2. Press “GO”

3. Enter the address location in which line program was executed

4. Press “Execute” key > **Result**

Thus 8085 microprocessor was studied successfully.

**8085** **INSTRUCTION** **SET**

Instructions can be categorized according to their method of addressing the hardware registers and/or memory.

> **Implied** **Addressing**

The addressing mode of certain instructions is implied by the instruction’s function. For example, the STC (set carry flag) instruction deals only with the carry flag, the DAA (decimal adjust accumulator) instruction deals with the accumulator.

> **Register** **Addressing**

Quite a large set of instructions are call for register addressing. With these instructions, you must specify one of the registers A through E, H or L as well as the operation code. With these instructions, the accumulator is implied as a second operand. For example, the instruction CMP E may be interpreted as 'compare the contents of the E register with the contents of the accumulator.

>**Immediate** **Addressing**

Instructions that use immediate addressing have data assembled as a part of the instruction itself. For example, the instruction CPI 'C' may be interpreted as ‘compare the contents of the accumulator with the letter C. When assembled, this instruction has the hexadecimal value FE43. Hexadecimal 43 is the internal representation for the letter C. When this instruction is executed, the processor fetches the first instruction byte and determines that it must fetch one more byte. The processor fetches the next byte into one of its internal registers and then performs the compare operation.

> **Direct** **Addressing**

Jump instructions include a 16-bit address as part of the instruction. For example, the instruction JMP 1000H causes a jump to the hexadecimal address 1000 by replacing the current contents of the program counter with the new value 1000H.Instructions that include a direct address require three bytes of storage: one for the instruction code, and two for the 16-bit address.

> **Register** **Indirect** **Addressing**

Register indirect instructions reference memory via a register pair. Thus, the instruction MOV M, C moves the contents of the C register into the memory address stored in the H and L register pair. The instruction LDAX B loads the accumulator with the byte of data specified by the address in the B and C register pair.

> **Combined** **Addressing** **Modes**

Some instructions use a combination of addressing modes. A CALL instruction, for example, combines direct addressing and registers indirect addressing. The direct address in a CALL instruction specifies the address of the desired subroutine; the register indirect address is the stack pointer. The CALL instruction pushes the current contents of the program counter into the memory location specified by the stack pointer.

>**Timing** **Effects** **of** **Addressing** **Modes**

Addressing modes affect both the amount of time required for executing an instruction and the amount of memory required for its storage. For example, instructions that use implied or register addressing, execute very quickly since they deal directly with the processor’s hardware or with data already present in hardware registers. Most important, however is that the entire instruction can be fetched with a single memory access. The number of memory accesses required is the single greatest factor in determining execution timing. More memory accesses therefore require more execution time. A CALL instruction for example, requires five memory accesses: three to access the entire instruction and two more to push the contents of the program counter onto the stack.

> **Instruction** **Naming** **Conventions**

The mnemonics assigned to the instructions are designed to indicate the function of the instruction. The instructions fall into the following functional categories:

**Data** **Transfer** **Group**

The data transfer instructions move data between registers or between memory and registers.

MOV Move

MVI Move Immediate

LDA Load Accumulator Directly from Memory STA Store Accumulator Directly in Memory

LHLD Load H & L Registers Directly from Memory SHLD Store H & L Registers Directly in Memory

An 'X' in the name of a data transfer instruction implies that it deals with a register pair(16- bits).

LXI Load Register Pair with Immediate data

LDAX Load Accumulator from Address in Register Pair STAX Store Accumulator in Address in Register Pair XCHG Exchange H & L with D & E

XTHL Exchange Top of Stack with H & L

**Arithmetic** **Group**

The arithmetic instructions add, subtract, increment, or decrement data in registers or memory.

ADD Add to Accumulator

ADI Add Immediate Data to Accumulator

ADC Add to Accumulator Using Carry Flag

ACI Add Immediate data to Accumulator Using Carry SUB Subtract from Accumulator

SUI Subtract Immediate Data from Accumulator

SBB Subtract from Accumulator Using Borrow (Carry) Flag

SBI Subtract Immediate from Accumulator Using Borrow (Carry) Flag INR Increment Specified Byte by One

DCR Decrement Specified Byte by One INX Increment Register Pair by One DCX Decrement Register Pair by One

DAD Double Register Add; Add Content of Register Pair to H & L Register Pai

**Logical** **Group**

This group performs logical (Boolean) operations on data in registers and memory and on condition flags.

The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the accumulator ON or OFF.

ANA Logical AND with Accumulator

ANI Logical AND with Accumulator Using Immediate Data ORA Logical OR with Accumulator

ORI Logical OR with Accumulator Using Immediate Data XRA Exclusive Logical OR with Accumulator

XRI Exclusive OR Using Immediate Data

The Compare instructions compare the content of an 8-bit value with the contents of the accumulator.

CMP Compare

CPI Compare Using Immediate Data

The rotate instructions shift the contents of the accumulator one bit position to the left or right.

RLC Rotate Accumulator Left RRC Rotate Accumulator Right RAL Rotate Left Through Carry RAR Rotate Right Through Carry

Complement and carry flag instructions:

CMA Complement Accumulator CMC Complement Carry Flag STC Set Carry Flag

**Branch** **Group**

The branching instructions alter normal sequential program flow, either unconditionally or conditionally. The unconditional branching instructions are as follows:

JMP Jump CALL Call RET Return

Conditional branching instructions examine the status of one of four condition flags to determine whether the specified branch is to be executed. The conditions that may be specified are as follows:

NZ Not Zero (Z = 0) Z Zero (Z = 1)

NC No Carry (C = 0) C Carry (C = 1)

PO Parity Odd (P = 0) PE Parity Even (P = 1)

P Plus (S = 0)

M Minus (S = 1)

Thus, the conditional branching instructions are specified as follows:

**Jumps** **Calls** JC CC JNC CNC JZ CZ JNZ CNZ JP CP JM CM JPE CPE

JP0 CPO

**Returns** RC (Carry)

RNC (No Carry) RZ (Zero)

RNZ (Not Zero) RP (Plus)

RM (Minus)

RPE (Parity Even)

RPO (Parity Odd)

Two other instructions can affect a branch by replacing the contents or the program counter:

PCHL Move H & L to Program Counter

RST Special Restart Instruction Used with Interrupts

> **Stack** **Control** **Instructions**

The following instructions affect the Stack and/or Stack Pointer:

PUSH Push Two bytes of Data onto the Stack POP Pop Two Bytes of Data off the Stack XTHL Exchange Top of Stack with H & L SPHL Move content of H & L to Stack Pointer

> **The** **I/O** **instructions**

IN Initiate Input Operation OUT Initiate Output Operation

> **Machine** **Control instruction**

EI Enable Interrupt System

DI Disable Interrupt System

HLT Halt

NOP No Operation

**LIST** **OF** **INSTRUCTIONS** **(WITH** **OPCODE** **AND** **DESCRIPTION)** **OF** **8085** **MICROPROCESSOR**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** **No.** | **Mnemonics,** **Operand** | **Opcode** | **Bytes** |
| 1. | ACI Data | CE | 2 |
| 2. | ADC A | 8F | 1 |
| 3. | ADC B | 88 | 1 |
| 4. | ADC C | 89 | 1 |
| 5. | ADC D | 8A | 1 |
| 6. | ADC E | 8B | 1 |
| 7. | ADC H | 8C | 1 |
| 8. | ADC L | 8D | 1 |
| 9. | ADC M | 8E | 1 |
| 10. | ADD A | 87 | 1 |
| 11. | ADD B | 80 | 1 |
| 12. | ADD C | 81 | 1 |
| 13. | ADD D | 82 | 1 |
| 14. | ADD E | 83 | 1 |
| 15. | ADD H | 84 | 1 |
| 16. | ADD L | 85 | 1 |
| 17. | ADD M | 86 | 1 |
| 18. | ADI Data | C6 | 2 |
| 19. | ANA A | A7 | 1 |
| 20. | ANA B | A0 | 1 |
| 21. | ANA C | A1 | 1 |
| 22. | ANA D | A2 | 1 |
| 23. | ANA E | A3 | 1 |
| 24. | ANA H | A4 | 1 |
| 25. | ANA L | A5 | 1 |
| 26. | ANA M | A6 | 1 |
| 27. | ANI Data | E6 | 2 |
| 28. | CALL Label | CD | 3 |
| 29. | CC Label | DC | 3 |
| 30. | CM Label | FC | 3 |
| 31. | CMA | 2F | 1 |
| 32. | CMC | 3F | 1 |
| 33. | CMP A | BF | 1 |
| 34. | CMP B | B8 | 1 |
| 35. | CMP C | B9 | 1 |
| 36. | CMP D | BA | 1 |
| 37. | CMP E | BB | 1 |
| 38. | CMP H | BC | 1 |
| 39. | CMP L | BD | 1 |
| 40. | CMP M | BD | 1 |
| 41. | CNC Label | D4 | 3 |
| 42. | CNZ Label | C4 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** **No.** | **Mnemonics,** **Operand** | **Opcode** | **Bytes** |
| 43. | CP Label | F4 | 3 |
| 44. | CPE Label | EC | 3 |
| 45. | CPI Data | FE | 2 |
| 46. | CPO Label | E4 | 3 |
| 47. | CZ Label | CC | 3 |
| 48. | DAA | 27 | 1 |
| 49. | DAD B | 09 | 1 |
| 50. | DAD D | 19 | 1 |
| 51. | DAD H | 29 | 1 |
| 52. | DAD SP | 39 | 1 |
| 53. | DCR A | 3D | 1 |
| 54. | DCR B | 05 | 1 |
| 55. | DCR C | 0D | 1 |
| 56. | DCR D | 15 | 1 |
| 57. | DCR E | 1D | 1 |
| 58. | DCR H | 25 | 1 |
| 59. | DCR L | 2D | 1 |
| 60. | DCR M | 35 | 1 |
| 61. | DCX B | 0B | 1 |
| 62. | DCX D | 1B | 1 |
| 63. | DCX H | 2B | 1 |
| 64. | DCX SP | 3B | 1 |
| 65. | DI | F3 | 1 |
| 66. | EI | FB | 1 |
| 67. | HLT | 76 | 1 |
| 68. | IN Port-address | DB | 2 |
| 69. | INR A | 3C | 1 |
| 70. | INR B | 04 | 1 |
| 71. | INR C | 0C | 1 |
| 72. | INR D | 14 | 1 |
| 73. | INR E | 1C | 1 |
| 74. | INR H | 24 | 1 |
| 75. | INR L | 2C | 1 |
| 76. | INR M | 34 | 1 |
| 77. | INX B | 03 | 1 |
| 78. | INX D | 13 | 1 |
| 79. | INX H | 23 | 1 |
| 80. | INX SP | 33 | 1 |
| 81. | JC Label | DA | 3 |
| 82. | JM Label | FA | 3 |
| 83. | JMP Label | C3 | 3 |
| 84. | JNC Label | D2 | 3 |
| 85. | JNZ Label | C2 | 3 |
| 86. | JP Label | F2 | 3 |
| 87. | JPE Label | EA | 3 |
| 88. | JPO Label | E2 | 3 |
| 89. | JZ Label | CA | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** **No.** | **Mnemonics,** **Operand** | **Opcode** | **Bytes** |
| 90. | LDA Address | 3A | 3 |
| 91. | LDAX B | 0A | 1 |
| 92. | LDAX D | 1A | 1 |
| 93. | LHLD Address | 2A | 3 |
| 94. | LXI B | 01 | 3 |
| 95. | LXI D | 11 | 3 |
| 96. | LXI H | 21 | 3 |
| 97. | LXI SP | 31 | 3 |
| 98. | MOV A, A | 7F | 1 |
| 99. | MOV A, B | 78 | 1 |
| 100. | MOV A, C | 79 | 1 |
| 101. | MOV A, D | 7A | 1 |
| 102. | MOV A, E | 7B | 1 |
| 103. | MOV A, H | 7C | 1 |
| 104. | MOV A, L | 7D | 1 |
| 105. | MOV A, M | 7E | 1 |
| 106. | MOV B, A | 47 | 1 |
| 107. | MOV B, B | 40 | 1 |
| 108. | MOV B, C | 41 | 1 |
| 109. | MOV B, D | 42 | 1 |
| 110. | MOV B, E | 43 | 1 |
| 111. | MOV B, H | 44 | 1 |
| 112. | MOV B, L | 45 | 1 |
| 113. | MOV B, M | 46 | 1 |
| 114. | MOV C, A | 4F | 1 |
| 115. | MOV C, B | 48 | 1 |
| 116. | MOV C, C | 49 | 1 |
| 117. | MOV C, D | 4A | 1 |
| 118. | MOV C, E | 4B | 1 |
| 119. | MOV C, H | 4C | 1 |
| 120. | MOV C, L | 4D | 1 |
| 121. | MOV C, M | 4E | 1 |
| 122. | MOV D, A | 57 | 1 |
| 123. | MOV D, B | 50 | 1 |
| 124. | MOV D, C | 51 | 1 |
| 125. | MOV D, D | 52 | 1 |
| 126. | MOV D, E | 53 | 1 |
| 127. | MOV D, H | 54 | 1 |
| 128. | MOV D, L | 55 | 1 |
| 129. | MOV D, M | 56 | 1 |
| 130. | MOV E, A | 5F | 1 |
| 131. | MOV E, B | 58 | 1 |
| 132. | MOV E, C | 59 | 1 |
| 133. | MOV E, D | 5A | 1 |
| 134. | MOV E, E | 5B | 1 |
| 135. | MOV E, H | 5C | 1 |
| 136. | MOV E, L | 5D | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** **No.** | **Mnemonics,** **Operand** | **Opcode** | **Bytes** |
| 137. | MOV E, M | 5E | 1 |
| 138. | MOV H, A | 67 | 1 |
| 139. | MOV H, B | 60 | 1 |
| 140. | MOV H, C | 61 | 1 |
| 141. | MOV H, D | 62 | 1 |
| 142. | MOV H, E | 63 | 1 |
| 143. | MOV H, H | 64 | 1 |
| 144. | MOV H, L | 65 | 1 |
| 145. | MOV H, M | 66 | 1 |
| 146. | MOV L, A | 6F | 1 |
| 147. | MOV L, B | 68 | 1 |
| 148. | MOV L, C | 69 | 1 |
| 149. | MOV L, D | 6A | 1 |
| 150. | MOV L, E | 6B | 1 |
| 151. | MOV L, H | 6C | 1 |
| 152. | MOV L, L | 6D | 1 |
| 153. | MOV L, M | 6E | 1 |
| 154. | MOV M, A | 77 | 1 |
| 155. | MOV M, B | 70 | 1 |
| 156. | MOV M, C | 71 | 1 |
| 157. | MOV M, D | 72 | 1 |
| 158. | MOV M, E | 73 | 1 |
| 159. | MOV M, H | 74 | 1 |
| 160. | MOV M, L | 75 | 1 |
| 161. | MVI A, Data | 3E | 2 |
| 162. | MVI B, Data | 06 | 2 |
| 163. | MVI C, Data | 0E | 2 |
| 164. | MVI D, Data | 16 | 2 |
| 165. | MVI E, Data | 1E | 2 |
| 166. | MVI H, Data | 26 | 2 |
| 167. | MVI L, Data | 2E | 2 |
| 168. | MVI M, Data | 36 | 2 |
| 169. | NOP | 00 | 1 |
| 170. | ORA A | B7 | 1 |
| 171. | ORA B | B0 | 1 |
| 172. | ORA C | B1 | 1 |
| 173. | ORA D | B2 | 1 |
| 174. | ORA E | B3 | 1 |
| 175. | ORA H | B4 | 1 |
| 176. | ORA L | B5 | 1 |
| 177. | ORA M | B6 | 1 |
| 178. | ORI Data | F6 | 2 |
| 179. | OUT Port-Address | D3 | 2 |
| 180. | PCHL | E9 | 1 |
| 181. | POP B | C1 | 1 |
| 182. | POP D | D1 | 1 |
| 183. | POP H | E1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** **No.** | **Mnemonics,** **Operand** | **Opcode** | **Bytes** |
| 184. | POP PSW | F1 | 1 |
| 185. | PUSH B | C5 | 1 |
| 186. | PUSH D | D5 | 1 |
| 187. | PUSH H | E5 | 1 |
| 188. | PUSH PSW | F5 | 1 |
| 189. | RAL | 17 | 1 |
| 190. | RAR | 1F | 1 |
| 191. | RC | D8 | 1 |
| 192. | RET | C9 | 1 |
| 193. | RIM | 20 | 1 |
| 194. | RLC | 07 | 1 |
| 195. | RM | F8 | 1 |
| 196. | RNC | D0 | 1 |
| 197. | RNZ | C0 | 1 |
| 198. | RP | F0 | 1 |
| 199. | RPE | E8 | 1 |
| 200. | RPO | E0 | 1 |
| 201. | RRC | 0F | 1 |
| 202. | RST 0 | C7 | 1 |
| 203. | RST 1 | CF | 1 |
| 204. | RST 2 | D7 | 1 |
| 205. | RST 3 | DF | 1 |
| 206. | RST 4 | E7 | 1 |
| 207. | RST 5 | EF | 1 |
| 208. | RST 6 | F7 | 1 |
| 209. | RST 7 | FF | 1 |
| 210. | RZ | C8 | 1 |
| 211. | SBB A | 9F | 1 |
| 212. | SBB B | 98 | 1 |
| 213. | SBB C | 99 | 1 |
| 214. | SBB D | 9A | 1 |
| 215. | SBB E | 9B | 1 |
| 216. | SBB H | 9C | 1 |
| 217. | SBB L | 9D | 1 |
| 218. | SBB M | 9E | 1 |
| 219. | SBI Data | DE | 2 |
| 220. | SHLD Address | 22 | 3 |
| 221. | SIM | 30 | 1 |
| 222. | SPHL | F9 | 1 |
| 223. | STA Address | 32 | 3 |
| 224. | STAX B | 02 | 1 |
| 225. | STAX D | 12 | 1 |
| 226. | STC | 37 | 1 |
| 227. | SUB A | 97 | 1 |
| 228. | SUB B | 90 | 1 |
| 229. | SUB C | 91 | 1 |
| 230. | SUB D | 92 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** **No.** | **Mnemonics,** **Operand** | **Opcode** | **Bytes** |
| 231. | SUB E | 93 | 1 |
| 232. | SUB H | 94 | 1 |
| 233. | SUB L | 95 | 1 |
| 234. | SUB M | 96 | 1 |
| 235. | SUI Data | D6 | 2 |
| 236. | XCHG | EB | 1 |
| 237. | XRA A | AF | 1 |
| 238. | XRA B | A8 | 1 |
| 239. | XRA C | A9 | 1 |
| 240. | XRA D | AA | 1 |
| 241. | XRA E | AB | 1 |
| 242. | XRA H | AC | 1 |
| 243. | XRA L | AD | 1 |
| 244. | XRA M | AE | 1 |
| 245. | XRI Data | EE | 2 |
| 246. | XTHL | E3 | 1 |

**LIST** **OF** **EXEPRIMENTS**

**ADDITION** **OF** **TWO** **8-BIT** **NUMBERS**

**Aim:** Write 8085 assembly language program for addition of two 8-bit numbers.

**Instruments** **Required**: 1. 8085 Microprocessor Kit 2. +5V Power supply

**Theory** **:** Consider the first number 42H is stored in memory location 8000H and the second number 35H is stored in memory location 8001H. The result after addition of two numbers is to be stored in the memory location 8002 H. Assume program starts from memory location 8500H.

**Algorithm**

1. Initialize the memory location of first number in HL register pair 2. Move first number/data into accumulator

3. Increment the content of HL register pair to initialize the memory location of second data 4. Add the second data with accumulator

5. Store the result in memory location 8003H **Program**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Machine*** ***Codes*** | ***Mnemonics*** | ***Comments*** |
| 8500 | 21 | LXI H, 8000 H | Address of first number in H-L register pair. |
| 8501 | 00 |
| 8502 | 80 |
| 8503 | 7E | MOV A,M | Transfer first number in accumulator. |
| 8504 | 23 | INX H | Increment content of H-L register pair |
| 8505 | 66 | ADD M | Add first number and second number |
| 8506 | 32 | STA 8003H | Store sum in 8003 H |
| 8507 | 03 |
| 8508 | 80 |
| 8509 | 76 | HLT | Halt |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Memory*** ***location*** | ***Data*** | ***Memory*** ***location*** | ***Data*** |
| 8000 | 42H | 8003 | 77H |
| 8001 | 35H |

**Calculation**

Data 1: 42 - 0100 0010 Data 2: 35 - 0011 0101 Sum: 77 – 01110111 Carry: 00

**Conclusion**

The addition of two 8-bit numbers is performed using 8085 microprocessor where sum is 8-bit.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What is the function of LXI H, 8000 H instruction? 2. How you can store a data in a memory location?

3. How you can read a data from a memory location? 4. What are flags available in 8085?

5. What is the function of RESET key of an 8085 microprocessor kit?

**SUBTRACTION** **OF** **TWO** **8** **BIT** **NUMBERS**

**Aim:** Write 8085 assembly language program for subtraction of two 8-bit numbers.

**Instruments** **Required**: 1. 8085 Microprocessor Kit 2. +5V Power supply

**Theory** **:** Consider the first number 55H is stored in memory location 8000H and the second number 32H is stored in memory location 8001H. The result after subtraction of two numbers is to be stored in the memory location 8002H. Assume program starts from memory location 8500H.

**Algorithm**

1. Initialize the memory location of first number in HL register pair 2. Move first number/data into accumulator

3. Increment the content of HL register pair to initialize the memory location of second data 4. Subtract the second data with accumulator

5. Store the result in memory location 8003H **Program**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Machine*** ***Codes*** | ***Mnemonics*** | ***Comments*** |
| 8500 | 21 | LXI H, 8000 H | Address of first number in H-L register pair. |
| 8501 | 00 |
| 8502 | 80 |
| 8503 | 7E | MOV A,M | Transfer first number in accumulator. |
| 8504 | 23 | INX H | Increment content of H-L register pair |
| 8505 | 66 | SUB M | Subtract first number and second number |
| ***Memory*** ***address*** | ***Machine*** ***Codes*** | ***Mnemonics*** | ***Comments*** |
| 8506 | 32 | STA 8003H | Store sum in 8003 H |
| 8507 | 03 |
| 8508 | 80 |
| 8509 | 76 | HLT | Halt |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Memory*** ***location*** | ***Data*** | ***Memory*** ***location*** | ***Data*** |
| 8000 | 55H | 8003 | 23H |
| 8001 | 32H |

**Calculation**

Data 1: 55 -0101 0101 Data 2: 32 -0011 0010 Difference: 23 -0010 0011 Borrow: 00

**Conclusion**

Subtraction of two 8-bit numbers is performed using 8085 microprocessor where sum is 8-bit. **Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What is the function of LXI H, 8000 H instruction? 2. How you can store a data in a memory location?

3. How you can read a data from a memory location? 4. What are flags available in 8085?

5. What is the function of RESET key of an 8085 microprocessor kit?

**ADDITION** **OF** **TWO** **16** **BIT** **NUMBERS**

**Aim:** To write an assembly language program for adding two 16 bit numbers using 8085 micro processor kit.

**Instruments** **required:** 1. 8085 Microprocessor Kit 2. +5V Power supply

**Theory:** Consider the first number 4283H is stored in memory location 8000H and 8001H; the second number 2931H is stored in memory location 8002H and 8003H. The result after addition of two numbers is to be stored in the memory location 8004H and 8005H. Assume program starts from memory location 8500H.

**Algorithm**

1. Start the microprocessor

2. Get the 1st 8 bit in ‘C’ register (LSB) and 2nd 8 bit in ‘H’ register (MSB) of 16 bit number. 3. Save the 1st 16 bit in ‘DE’ register pair

4. Similarly get the 2nd 16 bit number and store it in ‘HL’ register pair. 5. Get the lower byte of 1st number into ‘L’ register

6. Add it with lower byte of 2nd number 7. Store the result in ‘L’ register

8. Get the higher byte of 1st number into accumulator

9. Add it with higher byte of 2nd number and carry of the lower bit addition. 10. Store the result in ‘H’ register

11. Store 16 bit addition value in ‘HL’ register pair 12. Stop program execution

**Program**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | MVI | C,00 | 0E | C = 00H |
| 8501 |  |  |  | 00 |  |
| 8502 |  | LHLD | 8000 | 2A | HL – 1st No. |
| 8503 |  |  |  | 00 |  |
| 8504 |  |  |  | 80 |  |
| 8505 |  | XCHG |  | EB | HL – DE |
| 8506 |  | LHLD | 8002 | 2A | HL – 2nd No. |
| 8507 |  |  |  | 02 |  |
| 8508 |  |  |  | 80 |  |
| 8509 |  | DAD | D | 19 | Double addition DE + HL |
| 850A |  | JNC | Ahead | D2 | If Cy = 0, G0 to 850E |
| 850B |  |  |  | 0E |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 850C |  |  |  | 85 |  |
| 850D |  | INR | C | 0C | C = C + 01 |
| 850E | AHEAD | SHLD | 8004 | 22 | HL –8004 (sum) |
| 850F |  |  |  | 04 |  |
| 8510 |  |  |  | 80 |  |
| 8511 |  | MOV | C,A | 79 | Cy – A |
| 8512 |  | STA | 8006 | 32 | Cy – 8006 |
| 8513 |  |  |  | 06 |  |
| 8514 |  |  |  | 80 |  |
| 8515 |  | HLT |  | 76 | Stop execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Memory*** ***location*** | ***Value*** | ***Memory*** ***location*** | ***Value*** |
| 8000 | 83 (addend) | 8004 | B4 (sum) |
| 8001 | 42 (addend) | 8005 | 6B (sum) |
| 8002 | 31 (augend) | 8006 | 00 (carry) |
| 8003 | 29 (augend) |

**Calculation**

Lower Byte of Data1: (83)16 Lower Byte of Data2: (31)16 Lower Byte of Sum: (B4)16

Carry: 00

1000 0011 0011 0001

1011 0100

Higher Byte of Data1: (42)16 Higher Byte of Data2: (29)16

Higher Byte of Sum: (6B)16

0100 0010 0010 1001

0110 1011

**Conclusion**

Addition of two 16-bit numbers is performed using 8085 microprocessor where sum is 16-bit. **Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully. **Viva-Voice** **Questions**

1. What is the function of XCHG instruction?

2. How you can store a data in a memory location? 3. How you can read a data from a memory location? 4. What are flags available in 8085?

5. What is the function of DAD, LHLD and SHLD instructions of 8085 microprocessor?

**SUBTRACTION** **OF** **TWO** **16** **BIT** **NUMBERS**

**Aim:** To write an assembly language program for subtracting two 16 bit numbers using 8085 microprocessor kit.

**Instruments** **required:** 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** Consider the first number 4398H is stored in memory location 8000H and 8001H; the second number 4621H is stored in memory location 8002H and 8003H. The result after subtraction of two numbers is to be stored in the memory location 8004H and 8005H. Assume program starts from memory location 8500H.

**Algorithm**

1. Start the microprocessor

2. Get the 1st 16 bit in ‘HL’ register pair 3. Save the 1st 16 bit in DE register pair

4. Get the 2nd 16 bit number in HL register pair 5. Get the lower byte of 1st number

6. Get the subtracted value of 2nd number of lower byte by subtracting it with lower byte of 1st number

7. Store the result in ‘L’ register

8. Get the higher byte of 2nd number

9. Subtract the higher byte of 1st number from 2nd numbr with borrow 10. Store the result in ‘HL’ register

11. Stop the program execution

**Program**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | MVI | C,00 | 0E | C = 00H |
| 8501 |  |  |  | 00 |  |
| 8502 |  | LHLD | 8000 | 2A | L – 1st No. |
| 8503 |  |  |  | 00 |  |
| 8504 |  |  |  | 80 |  |
| 8505 |  | XLHG |  | EB | HL – DE |
| 8506 |  | LHLD | 8002 | 2A | HL – 2nd No. |
| 8507 |  |  |  | 02 |  |
| 8508 |  |  |  | 80 |  |
| 8509 |  | MOV | A,E | 7B | LSB of ‘1’ to ‘A’ |
| 850A |  | SUB | L | 95 | A – A – L |
| 850B |  | STA | 8004 | 32 | A – memory |
| 850C |  |  |  | 04 |  |
| 850D |  |  |  | 80 |  |
| 850E |  | MOV | A,D | 7A | MSB of 1 to A |
| 850F |  | SBB | H | 9C | A- A – H |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 8510 |  | STA | 8005 | 32 | A – memory |
| 8511 |  |  |  | 05 |  |
| 8512 |  |  |  | 80 |  |
| 8513 |  | HLT |  | 76 | Stop execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Memory*** ***location*** | ***Value*** | ***Memory*** ***location*** | ***Value*** |
| 8000 | 98 | 8004 | 22 |
| 8001 | 43 | 8005 | 55 |
| 8002 | 21 | 8006 | 00 (Borrow) |
| 8003 | 46 |

**Calculation**

Lower Byte of Data 1: 43 0100 0011 Lower Byte of Data 2: 21 0010 0001 Lower Byte of Difference: 22 00100010 Higher Byte of Data1: 98 1001 1000 Higher Byte of Data 2: 46 0100 0110 Higher Byte of Difference: 55 01010101 Borrow: 00

**Conclusion**

The subtraction of two 16-bit numbers is performed using 8085 microprocessor where result is 16-bit.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What is the function of XLHG instruction?

2. How you can store a data in a memory location? 3. How you can read a data from a memory location? 4. What are flags available in 8085?

5. What is the function of SBB, LHLD instructions of 8085 microprocessor?

**MULTIPLICATION** **OF** **TWO** **8** **BIT** **NUMBERS**

**Aim:** To write an assembly language for multiplying two 8 bit numbers by using 8085 micro processor kit.

**Instruments** **required:** 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** Consider the first number 03H is stored in accumulator directly; the second number 05H is stored in memory location 8001H. The result after multiplication of two numbers is to be stored in the memory location 8001H and 8002H. Assume program starts from memory location 8500H.

**Algorithm**

1. Start the microprocessor 2. Get the 1st 8 bit No.

3. Move the 1st 8it No. to a register

4. Get the 2nd 8 bit number

5. Move the 2nd 8 bit No. to another register

6. Initialize the accumulator as zero

7. Initialize the carry as zero

8. Add both register value accumulator

9. Jump on if no carry

10. Increment carries by 1 if there is

11. Decrement the 2nd value and repeat from step 8, till the 2nd value becomes zero. 12. Store the multiplied value in accumulator

13. Move the carry value to accumulator 14. Store the carry value in accumulator

**Program**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | MVI | A, 03 | 3E | A = 00H |
| 8501 |  |  |  | 03 |
| 8502 |  | MOV | E,A | 5F | E = A. |
| 8503 |  | MVI | D, 00 | 16 | Get the first number in DE register pair |
| 8504 |  |  |  | 00 |
| 8505 |  | LDA | 8000 | 3A | Store the content of memory location into A |
| 8506 |  |  |  | 00 |
| 8507 |  |  |  | 80 |
| 8508 |  | MOV | C,A | 4F | Initialize counter |
| 8509 |  | LXI | H, 0000 | 21 | Result = 0 |
| 850A |  |  |  | 00 |
| 850B |  |  |  | 00 |
| 850C | BACK | DAD | D | 19 | Result = Result + first number |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 850D |  | DCR | C | 0D | Decrement counter |
| 850E |  | JNZ | BACK | C2 | If count 0 repeat |
| 850F |  |  |  | 0C |
| 8510 |  |  |  | 85 |
| 8511 |  | SHLD | 8001 | 22 | Store result |
| 8512 |  |  |  | 01 |
| 8513 |  |  |  | 80 |
| 8514 |  | HLT |  | 76 | Stop execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Memory*** ***location*** | ***Value*** | ***Memory*** ***location*** | ***Value*** |
| 8000 | 05 | 8001 | 0F |
| Accumulator | 03 | 8002 | 00 |

**Calculation**

05 - 0000 0101 + 05 -0000 0101

----- 0A - 0000 1010 + 05 -0000 0101

= 0F - 0000 1111

**Conclusion**

The multiplication of two 8-bit numbers is performed using 8085 microprocessor where result is 16-bit.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What is the function of DAD instruction?

2. How you can store a data in a memory location? 3. How you can read a data from a memory location? 4. What are flags available in 8085?

5. What is the function of SHLD instruction of 8085 microprocessor?

**DIVISION** **OF** **TWO** **8** **BIT** **NUMBERS**

**Aim:** To write an assembly language program for dividing two 8 bit numbers using microprocessor kit.

**Instruments** **required:** 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** Consider the first number 09H is stored in memory location 8000H AND the second number 02H is stored in memory location 8001H. The result after division of two numbers is to be stored in the memory location 8002H (quotient) and 8003H (remainder). Assume program starts from memory location 8500H.

**Algorithm**

1. Start the microprocessor

2. Initialize the Quotient as 0

3. Load the 1st 8 bit data

4. Copy the contents of accumulator to register ‘B’ 5. Load the 2nd 8 bit data

6. Compare both the values

7. Jump if divisor is greater than dividend

1. Subtract the dividend value by divisor

9. Increment Quotient

10. Jump to step 7, till the dividend becomes zero 11. Store the result(Quotient) value in accumulator 12. Move the remainder value to accumulator

13. Store result in accumulator

14. Stop the program execution **Program**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | MVI | C, 00 | 0E | Initialize Quotient as zero |
| 8501 |  |  |  | 00 |
| 8502 |  | LDA | 8000 | 3A | Get the first number in Accumulator |
| 8503 |  |  |  | 00 |
| 8504 |  |  |  | 80 |
| 8505 |  | MOV | B,A | 47 | Copy the 1st data into register B |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | | ***Hex*** ***Code*** | ***Comments*** |
| 8506 |  | LDA | 8001 | 3A | Get the second number in Accumulator |
| 8507 |  |  |  | 01 |
| 8508 |  |  |  | 80 |
| 8509 |  | CMP | B | B8 | Compare the 2 values |
| 850A |  | JC | LOO P1 | DA | Jump if dividend lesser than divisor |
| 850B |  |  |  | 12 |
| 850C |  |  |  | 85 |
| 850D | LOOP2 | SUB | B | 90 | Subtract the 1st value by 2nd value |
| 850E |  | INR | C | 0C | Increment Quotient |
| 850F |  | JMP |  | C3 | Jump to Loop 1 till the value of dividend becomes zero |
| 8510 |  |  |  | 0D |
| 8511 |  |  |  | 85 |
| 8512 | LOOP1 | STA | 8002 | 32 | Store result |
| 8513 |  |  |  | 02 |
| 8514 |  |  |  | 80 |
| 8515 |  | MOV | A,C | 79 | Move the value of remainder to accumulator |
| 8516 |  | STA | 8003 | 32 | Store the remainder value in accumulator |
| 8517 |  |  |  | 03 |
| 8518 |  |  |  | 80 |
| 8519 |  | HLT |  |  | Stop execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Memory*** ***location*** | ***Value*** | ***Memory*** ***location*** | ***Value*** |
| 8000 | 09 | 8002 | 04 (quotient) |
| 8001 | 02 | 8003 | 01 (reminder) |

**Calculation**

1001 0010 – I ------0111 0010 – II ------0101

0010 – III

------0011

0010 – IV ------

0001 – carry

Quotient - 04 Carry - 01

**Conclusion**

The division of two 8-bit numbers is performed using 8085 microprocessor where result is 8 bit number quotient and 8 bit number remainder.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What is the function of CMP instruction?

2. How you can store a data in a memory location? 3. How you can read a data from a memory location? 4. What are flags available in 8085?

5. What is the function of JMP and JC instructions of 8085 microprocessor?

**ASCENDING** **ORDER**

**Aim:** To write a program to sort given ‘n’ numbers in ascending order

**Instruments** **required:** 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** A series of five words 04, AB, BC, 01,0A in memory locations from 8000 to 8004 and number of words is stored in memory location 17B3:0300. Arrange the above words in Descending Order.

**Algorithm**

1. Start the microprocessor

2. Accumulator is loaded with number of values to sorted and is saved 3. Decrement 8 register (N-1) Repetitions)

4. Set ‘HL’ register pair as data array

5. Set C register as counter for (N-1) repetitions 6. Load a data of the array in accumulator

7. Compare the data pointed in ‘HL’ pair

8. If the value of accumulator is smaller than memory, then jump to step 10. 9. Otherwise exchange the contents of ‘HL’ pair and accumulator

10. Decrement ‘C’ register, if the of ‘C’ is not zero go to step 6 11. Decrement ‘B’ register, if value of B is not zero, go step 3 12. Stop the program execution

**Program**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | LDA 8000 | 3A | Load the number of values |
| 8501 |  |  | 00 |
| 8502 |  |  | 80 |
| 8503 |  | MOV B,A | 47 | Move it ‘B’ register |
| 8504 |  | DCR B | 05 | For (N-1) comparisons |
| 8505 | Loop 3 | LXI H, 8000 | 21 | Set the pointer for array |
| 8506 |  |  | 00 |
| 8507 |  |  | 80 |
| 8508 |  | MOV C,M | 4E | Count for (N-1) comparisons |
| 8509 |  | DCR C | 0D | For (N-1) comparisons |
| 850A |  | INX H | 23 | Increment pointer |
| 850B | Loop 2 | MOV A,M | 7E | Get one data in array ‘A’ |
| 850C |  | INX H | 23 | Increment pointer |
| 850D |  | CMP M | BE | Compare next with accumulator |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 850E |  | JC Loop1 | DA | If content less memory go ahead |
| 850F |  |  | 16 |
| 8510 |  |  | 85 |
| 8511 |  | MOV D,M | 56 | If it is greater than interchange it |
| 8512 |  | MOV M,A | 77 | Memory content |
| 8513 |  | DCX H | 2B | Exchange the content of memory pointed by ‘HL’ by previous location |
| 8514 |  | MOV M,D | 72 | One in by ‘HL’ and previous location |
| 8515 |  | INX H | 23 | Increment pointer |
| 8516 | Loop 1 | DCR C | 0D | Decrement ‘C’ register |
| 8517 |  | JNZ Loop 1 | C2 | Repeat until ‘C’ is zero |
| 8518 |  |  | 0B |
| 8519 |  |  | 85 |
| 851A |  | DCR B | 05 | Decrement in ‘B’ values |
| 851B |  | JNZ Loop 2 | C2 | Repeat till ‘B’ is zero |
| 851C |  |  | 05 |
| 851D |  |  | 80 |
| 851E |  | HLT | 76 | Stop the program execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | 04 | 8000 | 04 |
| 8001 | AB | 8001 | 01 |
| 8002 | BC | 8002 | 0A |
| 8003 | 01 | 8003 | AB |
| 8004 | 0A | 8004 | BC |

**Conclusion**

The above assembly language program for sorting numbers in ascending order was executed by microprocessor kit and this program is stored into memory 8500 to 851E.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully. **Viva-Voice** **Questions**

1. What do you mean by ascending order?

2. What is the function of CMP, LXI instructions?

3. How you can store the smallest number in memory?

**LARGEST** **IN** **ARRAY**

**Aim:** To find the largest element in an array of size ‘n’ using 8085 Microprocessor.

**Instruments** **required:** 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** Find the largest number in a block of data. The length of the block is in memory location 8000H and the block itself starts from memory location 8001H. Store the maximum number in memory location 8050H. Assume that the numbers in the block are all 8 bit unsigned binary numbers.

**Algorithm**

1. Initialize counter

2. Maximum = Minimum possible value =0 3. Initialize pointer

4. Is number> maximum 5. Yes replace maximum

6. Decrement counter by one

7. Go to step 4 until counter= 0 8. Store maximum number

9. Terminate program execution

**Program**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | LDA 8000 | 3A | Load the number of values |
| 8501 |  |  | 00 |
| 8502 |  |  | 80 |
| 8503 |  | MOV C,A | 79 | Initialize counter |
| 8504 |  | XRA A | AF | Clear Accumulator |
| 8505 |  | LXI H, 8001 | 21 | Set the pointer for array |
| 8506 |  |  | 01 |
| 8507 |  |  | 80 |
| 8508 | BACK | CMP M | BD | Is number> maximum |
| 8509 |  | JNC SKIP | D2 | No, jump to SKIP |
| 850A |  |  | 0D |
| 850B |  |  | 85 |
| 850C |  | MOV A,M | 7E | replace maximum |
| 850D | SKIP | INX H | 23 | Increment pointer |
| 850E |  | DCR C | 0D | Decrement counter by one |
| 850F |  | JNZ BACK | C2 | Go to next iteration |
| 8510 |  |  | 08 |
| 8511 |  |  | 85 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 8512 |  | STA 8050 | 32 | Store maximum number |
| 8513 |  |  | 50 |
| 8514 |  |  | 80 |
| 8515 |  | HLT | 76 | Terminate program execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | 04 | 8050 | A9 |
| 8001 | 34 |  |
| 8002 | A9 |  |
| 8003 | 78 |  |
| 8004 | 56 |  |

**Conclusion**

Program to find the smallest element in an array of size ‘n’ using 8085 Microprocessor has been executed.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What do you mean by XRA A?

2. What is the function of CMP, LXI, JNC, JNZ instructions? 3. How you can store the largest number in memory?

**FIBONACCI** **SERIES**

**Aim:** To write an assembly language program to display ‘n’ elements of the Fibonacci series using 8085 Microprocessor.

**Instruments** **required:** 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** Find the Fibonacci series, where the length of the series is in memory location 8000H and the series itself starts from memory location 8001H.

**Algorithm**

1. Start the microprocessor

2. Load the length of series in the accumulator and decrement by 2 3. Move the value to register ‘D’

4. Load the starting value of data address

5. Initialize the 1st number as 00

6. Move the pointer to 2nd data and initialize as ‘01’

7. Move the pointer to next position for next data

8. Initialize B as 00 and C as ‘01’ for calculations

9. Copy the contents of ‘B’ to accumulator

10. Add the content of C register to accumulator 11. Move the content ‘C’ to ‘B’ and ‘A’ to C

12. Now store the result to memory pointed by HL pair 13. Move the pointer to next pointer

14. Decrement 0 by 1 for counter 15. If ‘D’ is not zero, go to step 9 16. If D is zero, end the program

**Program**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | LDA 8000 | 3A | Store the length of series in ‘A’ |
| 8501 |  |  | 00 |
| 8502 |  |  | 80 |
| 8503 |  | SUI 02 | D6 | Decrement ‘A’ by 02 |
| 8504 |  |  | 02 |
| 8505 |  | MOV D,A | 57 | Move ‘A’ to ‘D’ (counter) |
| 8506 |  | LXI H, 8001 | 21 | Load the starting address of array |
| 8507 |  |  | 01 |
| 8508 |  |  | 80 |
| 8509 |  | MVI M,00 | 36 | Initialize 8001 as ‘00’ |
| 850A |  |  | 00 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 850B |  | INX H | 23 | Increment pointer |
| 850C |  | MVI M, 01 | 36 | Initialize 2nd as ‘01’ |
| 850D |  |  | 01 |
| 850E |  | INX H | 23 | Increment pointer |
| 850F |  | MVI B,00 | 06 | Initialize ‘B’ as ‘00’ |
| 8510 |  |  | 00 |
| 8511 |  | MVI, C, 01 | 0E | Initialize ‘C’ as ‘01’ |
| 8512 |  |  | 01 |
| 8513 | Loop | MOV A,B | 78 | Move B to A |
| 8514 |  | ADD C | 81 | Add ‘A’ and ‘C’ |
| 8515 |  | MOV B,C | 41 | Move C to B |
| 8516 |  | MOV C,A | 4F | Move A to C |
| 8517 |  | MOV M,A | 77 | Move the result to memory |
| 8518 |  | INX H | 23 | Increment pointer |
| 8519 |  | DCR D | 15 | Decrement counter |
| 851A |  | JNZ Loop | C2 | If D = 0, jump to loop |
| 851B |  |  | 13 |
| 851C |  |  | 80 |
| 851D |  | HLT | 76 | Stop the program |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | 05 | 8001 | 00 |
| 8002 | 01 |
| 8003 | 01 |
| 8004 | 02 |
| 8005 | 03 |

**Conclusion**

The assembly language for Fibonacci series was executed successfully using 8085 microprocessor kit.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully. **Viva-Voice** **Questions**

1. What do you mean by SUI?

2. What is the function of LXI, JNC, JNZ instructions? 3. How you can store the series in memory?

**SUM** **OF** **SERIES** **OF** **DATA**

**Aim:** To write an assembly language program to calculate the sum of data using 8085 microprocessor kit.

**Instruments** **required:** 1.8085 Microprocessor Kit

2. +5V Power supply

**Theory:** This program finds the sum of series of data stored from 8001H on wards, where length of series is stored in memory location 8000H. The sum is stored in memory location 8050H and carry is stored in 8051H memory location.

**Algorithm**

1. Start the microprocessor

2. Load the number of values in series in accumulator and move it to register C and load the starting address of array

3. Initialize the value of A as ‘00’

4. Move the value of ‘A’ to ‘B’ register

5. Add the content of accumulator with the data pointed by HL pair 6. If there exists a carry, increment ‘B’ by 1, if not continue

7. Increment the pointer to next data

8. Decrement the value of C by 1, which is used as counter 9. If ‘C’ is equal to zero, go to step 10 if not go to step 5. 10. Store the value of ‘A’ to memory, it shows the result

11. Move the content of B to A

12. Store the value of A tomemory 13. Stop the program

**Program**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 8500 |  | LDA 8000 | 3A | Load accumulator with number of values |
| 8501 |  |  | 00 |
| 8502 |  |  | 80 |
| 8503 |  | MOV B,A | 4F | Move it from A to C |
| 8504 |  | LXI H, 8001 | 21 | Load the starting address of data array |
| 8505 |  |  | 01 |
| 8506 |  |  | 80 |
| 8507 |  | SUB A | 97 | Initialize ‘A’ as 00 |
| 8508 |  | MOV B,A | 47 | Initialize ‘B’ as 00 |
| 8509 | Loop | ADD M | 86 | Add the previous sum with next data |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***address*** | ***Label*** | ***Mnemonics*** | ***Hex*** ***Code*** | ***Comments*** |
| 850A |  | JNC Skip | D2 | Jump on if no carry |
| 850B |  |  | 0E |
| 850C |  |  | 85 |
| 850D |  | INR B | 04 | Increment carry by one |
| 850E | Skip | INX H | 23 | Increment pointer for next data |
| 850F |  | DCR C | 0D | Decrement ‘C’ by one |
| 8510 |  | JNZ Loop | C2 | Jump if not zero |
| 8511 |  |  | 09 |
| 8512 |  |  | 85 |
| 8513 |  | STA 8050 | 32 | Store the sum in accumulator |
| 8514 |  |  | 50 |
| 8515 |  |  | 80 |
| 8516 |  | MOV A,B | 78 | Move the value of carry to A from B |
| 8517 |  | STA 8051 | 32 | Store the carry in memory |
| 8518 |  |  | 51 |
| 8519 |  |  | 80 |
| 851A |  | HLT | 76 | End of program |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | 04 | 8050 | 17 |
| 8001 | 07 | 8051 | 00 |
| 8002 | 09 |  |
| 8003 | 03 |  |
| 8004 | 04 |  |

**Calculation**

07 + 09 + 03 + 04 = 23 = 17(in Hexa decimal) (0F + 8 =233)

0F = 0000 1111 08 = 0000 1000 ---------------0001 0111

**Conclusion**

The assembly language program for sum of data was executed successfully using 8085 microprocessor kit.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What do you mean by carry?

2. What is the function of LXI, JNC, JNZ instructions? 3. How you can store the series in memory?

**FACTORIAL** **OF** **8** **BIT** **NUMBER** **USING** **SUBROUTINE**

**Aim:** To write a program to calculate the factorial of a number (between 0 to 8)

**Instruments** **required:** 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** This program finds the factorial of a number stored in 8000H memory location. The result is stored in memory location 8050H.

**Algorithm:**

1. Initialize the stack pointer

2. Get the number in accumulator

3. Check for if the number is greater than 1. If no store the result otherwise go to next step. 4. Load the counter and initialize result

5. Now factorial program in sub-routine is called.

6. In factorial,initialize HL RP with0. Move the count value to B 7. Add HL content with Rp.

8. Decrement count (for multiplication) 9. Exchange content of Rp(DE) with HL.

10. Decrement counter (for factorial)till zero flag is set. 11. Store the result

12. Halt

**Program**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***Address*** | ***Label*** | ***Hex*** ***Code*** | ***Mnemonics*** | ***Comments*** |
| 8500 8501 8502 |  | 3A 00 80 | LDA 8000 | Get the number in accumulator |
| 8503 8504 |  | FE 02 | CPI 02H | Compare data with 2 and check it is greater than 1 |
| 8505 8506 8507 |  | DA 17 85 | JC Loop 1 | If cy =1 jump to loop 1 If cy = 0 proceed |
| 8508 |  | 5F | MOV E,A | Move content of A to E |
| 8509 850A |  | 16 00 | MVI D,00 | Load this term as a result |
| 850B |  | 3D | DCR A | Decrement accumulator by 1 |
| 850C |  | 4F | MOV C,A | Move ‘A’ content to ‘C’ (counter 1 less than A) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***Address*** | ***Label*** | ***Hex*** ***Code*** | ***Mnemonics*** | ***Comments*** |
| 850D 850E 850F |  | CD 00 86 | CALL Facto | Call sub routine program Facto |
| 8510 |  | EB | XCHG | Exchange (DE) – (HL) |
| 8511 8512 8513 |  | 22 50 80 | SHLD 8050 | Store content of HL in specified memory location |
| 8514 8515 8516 |  | C3 1D 85 | JMP Loop 3 | Jump to Loop 3 |
| 8517 8518 8519 | Loop1 | 21 00 01 | LXI H,0001H | HL is loaded with data 01 |
| 851A 851B 851C |  | 22 50 80 | SHLD 8050 | Store the result in memory |
| 851D | Loop3 | 76 | HLT | Terminate the program |
| **Sub** **Routine** | | | | |
| ***Memory*** ***Address*** | ***Label*** | ***Hex*** ***Code*** | ***Mnemonics*** | ***Comments*** |
| 8600 8601 8602 | Facto | 21 00 00 | LXI H,0000 | Initialize HL pair |
| 8603 |  | 41 | MOV B,C | Content of ‘C’ is moved to B |
| 8604 | Loop2 | 19 | DAD D | Content of DE is added with HL |
| 8605 |  | 05 | DCR B | ‘B’ is decremented |
| 8606 8607 8608 |  | C2 04 86 | JNZ Loop 2 | Multiply by successive addition till zero flag is set |
|  |  | EB | XCHG | [DE] – [HL] |
|  |  | 0D | DCR C | Decrement counter value |
|  |  | C4 00 46 | CNZ Facto | Call on no zero to facto (i.e repeat process till zero flag for c = 1) |
|  |  | C9 | RET | Return to main program |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | 04 | 8050 | 18 |

**Conclusion**

The assembly language program for factorial of 8 bit number was executed successfully using 8085 microprocessor kit.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions**

1. What do you mean by CALL?

2. What is the function of SHLD, CNZ instructions? 3. How subroutine is executed?

**1’s** **COMPLEMENT** **OF** **AN** **8** **BIT** **NUMBER**

**Aim:** Write 8085 assembly language program for one’s complement of an 8-bit numbers **Instruments** **Required**: 1.8085 Microprocessor Kit

2. +5V Power supply

**Theory:** The number is stored in memory location 8050H and one’s complement of number will be stored in location 8051H. Assume the program memory starts from 8000H.

**Algorithm**

1. Load memory location of data 8000H in H-L registers pair. 2. Move data into accumulator

3. Complement accumulator

4. Store the result in memory location 8050H **Program**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Memory*** ***Address*** | ***Hex*** ***Code*** | ***Mnemonics*** | ***Comments*** |
| 8500 | 21 | LXI H,8000H | Load address of number in H-L register pair |
| 8501 | 00 |  |
| 8502 | 80 |  |
| 8503 | 7E | MOV A,M | Move number into accumulator |
| 8504 | 3F | CMA | Complement accumulator |
| 8505 | 32 | STA 8050H | Store the result in 8050H |
| 8506 | 50 |  |
| 8507 | 80 |  |
| 8508 | 76 | HLT | Stop Execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | F0H | 8050 | 0FH |

**Conclusion**

The one’s complement of an 8-bit numbers is performed using 8085 microprocessor**.** **Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully. **Viva-Voice** **Questions:**

1. Define one’s complement of an 8-bit numbers. 2. What is the function of CMA instruction?

**2’s** **COMPLEMENT** **OF** **AN** **8** **BIT** **NUMBER**

**Aim:** Write 8085 assembly language program for two’s complement of an 8-bit numbers

**Instruments** **Required**: 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** The number is stored in memory location 8000H. The two’s complement will be stored in 8050H. The program is written from memory location 8500H.

**Algorithm**

1. Transfer the content of memory location 8500H to accumulator. 2. Complement the content of accumulator

3. Add 01H with accumulator to get two’s complement of number

4. Store the result in memory location 8501H

**Program**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Memory*** ***Address*** | ***Hex*** ***Code*** | ***Mnemonics*** | ***Comments*** |
| 8500 | 21 | LXI H,8000H | Load address of number in H-L register pair |
| 8501 | 00 |  |
| 8502 | 80 |  |
| 8503 | 7E | MOV A,M | Move number into accumulator |
| 8504 | 3F | CMA | Complement accumulator |
| 8505 | C6 | ADI 01 | Add 01H with accumulator to find two’s complement of number |
| 8506 | 01 | 01 |
| 8507 | 32 | STA 8050H | Store the result in 8050H |
| 8508 | 50 |  |
| 8509 | 80 |  |
| 850A | 76 | HLT | Stop Execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | F0H | 8050 | 10H |

**Conclusion:**

The two’s complement of an 8-bit numbers is performed using 8085 microprocessor.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions:**

1. Define two’s complement of an 8-bit numbers. 2. What is the function of CMA instruction?

3. Why ADI 01H is used in two’s complement of an 8-bit number.

**BLOCK** **TRANSFER** **PROGRAM**

**Aim:** Write 8085 assembly language program to transfer block of 16 bytes data from source to destination.

**Instruments** **Required**: 1.8085 Microprocessor Kit 2. +5V Power supply

**Theory:** The block of 16 byte data stored in memory locations 8000H onwards and transfers this data to another memory locations and store from 8050 onwards.

**Algorithm**

1. Initialize counter

2. Initialize source memory pointer

3. Initialize destinationMemory pointer 4. Get byte from source memory block

5. Store byte in the destinationMemory block 6. Decrement source memory pointer

7. Decrement destinationMemory pointer 8. Decrement counter

9. If counter0 repeat 10. Stop execution

**Program**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***Address*** | ***Hex*** ***Code*** | ***Label*** | ***Mnemonics*** | ***Comments*** |
| 8500 | 21 |  | LXI H,8000H | Load address of number in H-L register pair |
| 8501 | 00 |  |  |
| 8502 | 80 |  |  |
| 8503 | 11 |  | LXI D,8050H | Load address of number in D-E register pair |
| 8504 | 50 |  |  |
| 8505 | 80 |  |  |
| 8506 | 0E |  | MVI C, 0F | Initialize the counter C |
| 8507 | 0F |  |  |
| 8508 | 7E | Loop | MOV A,M | Move number into accumulator |
| 8509 | 12 |  | STAX D | Data transfer from source to destination |
| 850A | C6 |  | INX H | Increment source pointer |
| 850B | 01 |  | INX D | Increment destination pointer |
| 850C | 0D |  | DCR C | Decrement counter |
| 850D | C2 |  | JNZ Loop | Go to the loop staring location if z= 0 |
| 850E | 08 |  |  |
| 850F | 85 |  |  |
| 8510 | 76 |  | HLT | Stop Execution |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | 00 | 8050 | 00 |
| 8001 | 01 | 8051 | 01 |
| 8002 | 02 | 8052 | 02 |
| 8003 | 03 | 8053 | 03 |
| 8004 | 04 | 8054 | 04 |
| 8005 | 05 | 8055 | 05 |
| 8006 | 06 | 8056 | 06 |
| 8007 | 07 | 8057 | 07 |
| 8008 | 08 | 8058 | 08 |
| 8009 | 09 | 8059 | 09 |
| 800A | 0A | 805A | 0A |
| 800B | 0B | 805B | 0B |
| 800C | 0C | 805C | 0C |
| 800D | 0D | 805D | 0D |
| 800E | 0E | 805E | 0E |
| 800F | 0F | 805F | 0F |

**Conclusion:**

The block transfer is performed using 8085 microprocessor.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions:** 1. Define block transfer.

2. What is the function of STAX instruction? 3. Why INX H and INX D is used?

**SUM** **OF** **TWO** **BCD** **NUMBERS**

**Aim:** To perform addition of two 8-bit BCD numbers using 8085 microprocessor.

**Instruments** **Required**: 1. 8085 Microprocessor Kit 2. +5V Power supply

**Theory:** The two numbers in BCD are stored in memory locations 8000H and 8001H respectively. This program adds these two BCD numbers and stores the results in memory locations 8003H and 8004H.

**Algorithm**

1. Load Data 1 into Accumulator

2. Move Accumulator contents to B 3. Load Data 2 into accumulator

4. Add Data1 and Data2 and store into Accumulator 5. Convert the accumulator value to BCD value

6. Store Accumulator content (Result) to memory

**Program**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***Address*** | ***Hex*** ***Code*** | ***Label*** | ***Mnemonics*** | ***Comments*** |
| 8500 | 3A |  | LDA 8000 | Load Data 1 into Accumulator |
| 8501 | 00 |  |  |
| 8502 | 80 |  |  |
| 8503 | 47 |  | MOV B,A | Move Accumulator contents to B |
| 8504 | 3A |  | LDA 8001 | Load Data 2 into accumulator |
| 8505 | 01 |  |  |
| 8506 | 80 |  |  |
| 8507 | 0E |  | MVI C, 00 | Clear C to account for Carry |
| 8508 | 00 |  |  |
| 8509 | 86 |  | ADD B | Add Data 2 to Data 1 and store in Accumulator |
| 850A | 27 |  | DAA | Convert the accumulator value to BCD value |
| 850B | D2 |  | JNC Next | If carry==0 , go to Next |
| 850C | 0F |  |  |
| 850D | 85 |  |  |
| 850E | 0C |  | INR C | If carry==1, Increment C by 1 |
| 850F | 32 | Next | STA 8003 | Store Accumulator content (Result) to memory |
| 8510 | 03 |  |  |
| 8511 | 80 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Memory*** ***Address*** | ***Hex*** ***Code*** | ***Label*** | ***Mnemonics*** | ***Comments*** |
| 8512 | 79 |  | MOV A,C | Move contents of C to Accumulator |
| 8513 | 32 |  | STA 8004 | Store Register C (Carry) content to memory |
| 8514 | 04 |  |  |
| 8515 | 80 |  |  |
| 8516 | 76 |  | HLT | End of program |

**Experimental** **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Input*** ***Data*** | | ***Result*** | |
| ***Input*** ***Address*** | ***Value*** | ***Output*** ***Address*** | ***Value*** |
| 8000 | 13 (Input1) | 8003 | 63 |
| 8001 | 4A (Input2) | 8004 | 0 |

**Calculation**

Input 1: 13 0001 0011 Input22: 4A 0100 1010 Gives 0101 1101

After adding 06 0000 0110 Result: 63 0110 0011 Carry: 00

**Note:** If hexadecimal number is in units place 06 is added, if hexadecimal is in tens place 60 is added to sum by the DAA command.

**Conclusion:**

The BCD addition is performed using 8085 microprocessor successfully.

**Precautions**

1. Properly connect the 8085 microprocessor kit with power supply terminals. 2. Switch on the power supply after checking connections.

3. Handle the Trainer kit carefully.

**Viva-Voice** **Questions:**

1. How BCD number store in memory?

2. What is the function of DAA instruction? 3. How you can store the results in memory?