**MERGE SORT PROGRAM:**

Assumption: 8 numbers are to be sorted.

Note: This program can be scaled for any amount of data just by increasing n and adding more CALL MERGE commands with relevant initial conditions.

**LXI H,#0F000**H// Load address to HL pair containing N=8

**MOV A,M** // Put N in accumulator

**RAR** // To find half of the total numbers

**MOV C,A** // Put accumulator content in the C register

**INX H** // Increment HL pair to next address #0F001

**BACK**: **MOV A,M** // Put memory content pointed out by HL pair in //accumulator

**INX H** // Increment HL pair

**CMP M** // Compare accumulator content with the content in //memory

**JC SKIP** // Jump to SKIP if Carry Flag is SET

**MOV B,M** // Put memory content pointed out by HL pair in B //register

**MOV M,A** // Put accumulator content in memory location //pointed by HL register pair

**DCX H** // Decrement HL pair

**MOV M,B** // Put content of register B in memory location //pointed by HL register pair

**INX H** // Increment HL pair

**SKIP**: **INX H** // Increment HL pair

**DCR C** // Decrement C register

**JNZ BACK** // Jump to Back Loop as long as ZERO flag is not //set

**LXI SP,#0D000H** // Initialize Stack Pointer to empty address //which will act as temporary stack to store data while rearrangement is going //on.

**LXI H,#0C000H** // Initialize HL pair with address C000H

**PUSH H //** Push the address which is in HL register pair to top of //the stack pointed by SP

**LXI H,#0F001H** // Initialize HL pair as pointer to first pair of //data

**LXI D,#0F003H** //Initialize DE pair as pointer to second pair of //data

**MVI B,#002H** // B register will act as a counter ; 02 because //there are two numbers in first pair

**MVI C,#002H** // C register will also act as a counter

**MVI A,#02FH** // Move address of CALL MERGE command to //accumulator. As the program has been written from address #00000H, we //only need to move the LSB address.

**STA #0FF00H //** Store accumulator content to address for later //access

**CALL MERGE** // Calling function MERGE

**LXI SP,#0D000H** // Initialize Stack Pointer to empty address //which will act as temporary stack to store data while rearrangement is going //on.

**LXI H,#0C004H** // Initialize HL pair with address C004H

**PUSH H //** Push the address which is in HL register pair to top of //the stack pointed by SP

**LXI H,#0F005H** // Initialize HL pair as pointer to third pair of

**//**data

**LXI D,#0F007H** // Initialize DE pair as pointer to fourth pair of //data

**MVI B,#002H** // B register will act as counter

**MVI C,#002H** // C register will act as counter

**MVI A,#48FH** // Move address of CALL MERGE command to //accumulator. As the program has been written from address #00000H, we //only need to move the LSB address.

**STA #0FF00H //** Store accumulator content to address for later //access

**CALL MERGE** // Calling Merge function

**LXI SP,#0D000H** // Initialize Stack Pointer to empty address //which will act as temporary stack to store data while rearrangement is going //on.

**LXI H,#0F001H** // Initialize HL pair with address F001H

**PUSH H //** Push the address which is in HL register pair to top of //the stack pointed by SP

**LXI H,#0C000H** // Initialize HL pair as pointer to third pair of

**//**data

**LXI D,#0C004H** // Initialize DE pair as pointer to fourth pair of

//data

**MVI B,#004H** // B register will act as counter

**MVI C,#004H** // C register will act as counter

**MVI A,#061FH** // Move address of CALL MERGE command to //accumulator. As the program has been written from address #00000H, we //only need to move the LSB address.

**STA #0FF00H //** Store accumulator content to address for later //access

**CALL MERGE** // Calling Merge function

**HLT**

**SUBROUTINE**

**MERGE**: **MOV A,M** // Move memory location content to accumulator

**XCHG** // Exchange HL 🡨🡪 DE

**CMP M** // Compare accumulator content with memory content. //As HL pair is now pointing where DE was pointing earlier, hence two //different sets’ elements are being compared.

**JNC BIG** // Jump to BIG if Carry flag in not SET

**LXI SP,#0CFFEH** // Initialize Stack Pointer to top of the stack //where the address of the new stack will be available where we need to put //the sorted data

**XTHL** // Exchange HL🡨🡪[SP]

**MOV M,A** // Move accumulator content to memory

**INX H** // Increment HL pair

**XTHL** // Exchange HL 🡨🡪 [SP]

**INX D** // Increment DE pair

**XCHG** // Exchange HL 🡨🡪DE

**DCR B** // Decrement counter B

**JNZ MERGE** // Jump to Merge if Carry Flag not SET

**XCHG** // Exchange HL🡨🡪DE

**LOOP\_B**: **MOV B,M** // If all elements of first set are already moved and //exhausted, then we just need to move all elements of other set to the stack //below them. LOOP\_B does this task of moving what’s left of other set to //the stack.

**LXI SP,#0CFFEH** // Initialize Stack Pointer to top of the stack //where the address of the new stack will be available where we need to put //the sorted data

**XTHL** // Exchange HL🡨🡪SP

**MOV M,B** // Move content of B register to memory location

**INX H** // Increment HL pair

**XTHL** // Exchange HL🡨🡪[SP]

**INX H** // Increment HL pair

**DCR C** // Decrement C register which is acting as a counter

**JNZ LOOP\_B** // Jump to LOOP\_B if ZERO flag NOT SET

**MVI H,#000H** // Initialize H register with value #000H

**LDA #0FF00H** // Load accumulator with value stored at address #0FF00H

**MOV L,A** // Move accumulator content to L register.

**XTHL** // Exchange HL🡨🡪[SP]

**RET** // Return to main program

**BIG**: **MOV A,M** // Move memory location content to accumulator

**LXI SP,#0CFFEH** // Initialize Stack Pointer to top of the stack //where the address of the new stack will be available where we need to put //the sorted data

**XTHL** // Exchange HL🡨🡪[SP]

**MOV M,A** // Move accumulator content to memory location

**INX H** // Increment HL pair

**XTHL** // Exchange HL🡨🡪[SP]

**INX H** // Increment HL Pair

**XCHG** // Exchange HL🡨🡪DE

**DCR C** // Decrement C register

**JNZ MERGE** // Jump to Merge if ZERO flag NOT SET

**LOOP\_A**: **MOV C,M** // Move memory content to C register

**LXI SP,#0CFFEH** // Initialize Stack Pointer to top of the stack //where the address of the new stack will be available where we need to put //the sorted data

**XTHL** // Exchange HL🡨🡪[SP]

**MOV M,C** // Move content of C register to memory location

**INX H** // Increment HL Pair

**XTHL** // Exchange HL🡨🡪[SP]

**INX H** // Increment HL Pair

**DCR B** // Decrement B register

**JNZ LOOP\_A** // Jump to LOOP\_A if ZERO flag NOT SET

**MVI H,#000H** // Initialize H register with value #000H

**LDA #0FF00H** // Load accumulator with value stored at address #0FF00H

**MOV L,A** // Move accumulator content to L register.

**XTHL** // Exchange HL🡨🡪[SP]

**RET** // Return to main program