1. **Sum of two numbers and store it in the memory**

1. **Sum of two 8-bit number with carry**

MVI A,A5H

STA 8000H

MVI A, 87H

STA 8001H

LDA 8000

MOV B,A

LDA 8001

ADD B

JNC SKIP

INR C

SKIP: STA 8002H

MOV A,C

STA 8003H

RST 5

1. **Multiply two 8-bit number using successive addition**

1. **Division of two 8-bit numbers**

MVI D,14H

MVI E,07H

MVI B,00H

MOV A,D

LOOP: CMP E

JC SKIP

SUB E

INR B

JMP LOOP

SKIP: STA 8000H

MOV A,B

STA 8001H

1. Sort the stored number in memory
2. **Factorial of a number**

LXI H,3000

  MOV B,M

  MVI D,01

LOOP2:   CALL MUL

  DCR B

  JNZ LOOP2

  LXI H,4000

  MOV M,D

  HLT

MUL:   PUSH PSW

  MOV C,B

  XRA A

LOOP1:   ADD D

  DCR C

  JNZ LOOP1

  MOV D,A

  POP PSW

  RET

1. **Even or Odd**

MVI A, 07H

STA 2000H

LXI H, 2000H

    MOV A,M

    ANI 01H

    JZ EVEN

ODD:

    LXI H, 3000H

    MVI M, 01H

    JMP END

EVEN:

    LXI H, 3000H

    MVI M, 02H

END:

    HLT

1. **Even parity in any number**

   LXI H, 3000H

    MVI M, 6BH

    MOV A,M

    MVI B, 08H

    MVI C, 00H

CHECK\_BITS:

    RRC

    JNC SKIP\_INCR

    INR C

SKIP\_INCR:

    DCR B

    JNZ CHECK\_BITS

    MOV A,C

    ANI 01H

    JZ EVEN\_PARITY

ODD\_PARITY:

    LXI H, 3001H

    MVI M, 01H

    JMP END

EVEN\_PARITY:

    LXI H, 3001H

    MVI M, 02H

END:

    HLT

1. **Copy data from 2050-2055 to 2060-2065**

  LXI H,2050 // Load HL with the start address 2050H

  MVI A,56 // Load A with data 01H

  MOV M,A // Store 01H at 2050H

  INX H // Increment HL to point to 2051H

  MVI A,2D // Load A with data 02H

  MOV M,A // Store 02H at 2051H

  INX H // Increment HL to point to 2052H

  MVI A,3E // Load A with data 03H

  MOV M,A // Store 03H at 2052H

  INX H // Increment HL to point to 2053H

  MVI A,04 // Load A with data 04H

  MOV M,A // Store 04H at 2053H

  INX H // Increment HL to point to 2054H

  MVI A,0A // Load A with data 05H

  MOV M,A // Store 05H at 2054H

  INX H // Increment HL to point to 2055H

  MVI A,17 // Load A with data 06H

  MOV M,A // Store 06H at 2055H

  LXI H,2050

  LXI D,2060

COPY\_LOOP:   MOV A,M

  STAX D

  INX H

  INX D

  DCR C

  JNZ COPY\_LOOP

  HLT

1. **“ Paste reversely**

  LXI H,2050 // Load HL with the start address 2050H

  MVI A,01 // Load A with data 01H

  MOV M,A // Store 01H at 2050H

  INX H // Increment HL to point to 2051H

  MVI A,02 // Load A with data 02H

  MOV M,A // Store 02H at 2051H

  INX H // Increment HL to point to 2052H

  MVI A,03 // Load A with data 03H

  MOV M,A // Store 03H at 2052H

  INX H // Increment HL to point to 2053H

  MVI A,04 // Load A with data 04H

  MOV M,A // Store 04H at 2053H

  INX H // Increment HL to point to 2054H

  MVI A,05 // Load A with data 05H

  MOV M,A // Store 05H at 2054H

  INX H // Increment HL to point to 2055H

  MVI A,06 // Load A with data 06H

  MOV M,A // Store 06H at 2055H

  LXI H,2055

  LXI D,2060

REVERSE\_COPY\_LOOP:   MOV A,M

  STAX D

  DCX H

  INX D

  DCR C

  JNZ REVERSE\_COPY\_LOOP

  HLT

1. **“ Exchange data**

  LXI H,2050 // Load HL with the start address 2050H

  MVI A,01 // Load A with data 01H

  MOV M,A // Store 01H at 2050H

  INX H // Increment HL to point to 2051H

  MVI A,02 // Load A with data 02H

  MOV M,A // Store 02H at 2051H

  INX H // Increment HL to point to 2052H

  MVI A,03 // Load A with data 03H

  MOV M,A // Store 03H at 2052H

  INX H // Increment HL to point to 2053H

  MVI A,04 // Load A with data 04H

  MOV M,A // Store 04H at 2053H

  INX H // Increment HL to point to 2054H

  MVI A,05 // Load A with data 05H

  MOV M,A // Store 05H at 2054H

  INX H // Increment HL to point to 2055H

  MVI A,06 // Load A with data 06H

  MOV M,A // Store 06H at 2055H

// Initialize data in memory locations 2060H-2065H

  LXI H,2060 // Load HL with the start address 2060H

  MVI A,10 // Load A with data 10H

  MOV M,A // Store 10H at 2060H

  INX H // Increment HL to point to 2061H

  MVI A,20 // Load A with data 20H

  MOV M,A // Store 20H at 2061H

  INX H // Increment HL to point to 2062H

  MVI A,30 // Load A with data 30H

  MOV M,A // Store 30H at 2062H

  INX H // Increment HL to point to 2063H

  MVI A,40 // Load A with data 40H

  MOV M,A // Store 40H at 2063H

  INX H // Increment HL to point to 2064H

  MVI A,50 // Load A with data 50H

  MOV M,A // Store 50H at 2064H

  INX H // Increment HL to point to 2065H

  MVI A,60 // Load A with data 60H

  MOV M,A // Store 60H at 2065H

// Swap data between 2050H-2055H and 2060H-2065H

  LXI H,2050 // Load HL with the start address of the first block (2050H)

  LXI D,2060 // Load DE with the start address of the second block (2060H)

// Set the counter to 6 (number of bytes to swap)

SWAP\_LOOP:   MOV A,M // Load data from source address (HL) into A

  MOV B,A // Store the data in B temporarily

  LDAX D // Load data from destination address (DE) into A

  MOV M,A // Store data from DE into the HL location

  MOV A,B // Retrieve original data from B to A

  STAX D // Store it in the DE location

  INX H // Increment HL to point to the next address in the first block

  INX D // Increment DE to point to the next address in the second block

  DCR C // Decrement the counter

  JNZ SWAP\_LOOP // If counter is not zero, repeat the loop

  HLT // Halt the program

1. **Fibonacci series**

LXI SP,FFFF

MVI C,10

LXI H,0000

LXI D,0001

PUSH H

PUSH D

LOOP: DAD D

POP D

PUSH D

PUSH H

DCR C

JNZ LOOP

HLT

1. **Largest number**

  LXI H,2050 // Load HL with the start address 2050H

  MVI A,12 // Load A with the value 12H

  MOV M,A // Store 12H at 2050H

  INX H // Increment HL to point to 2051H

  MVI A,34 // Load A with the value 34H

  MOV M,A // Store 34H at 2051H

  INX H // Increment HL to point to 2052H

  MVI A,56 // Load A with the value 56H

  MOV M,A // Store 56H at 2052H

  INX H // Increment HL to point to 2053H

  MVI A,78 // Load A with the value 78H

  MOV M,A // Store 78H at 2053H

  INX H // Increment HL to point to 2054H

  MVI A,9A // Load A with the value 9AH

  MOV M,A // Store 9AH at 2054H

// Finding the smallest number in 2050H - 2054H

  LXI H,2050 // Load HL with the start address of the array (2050H)

  MOV A,M // Load the first element into A

  MOV B,A // Copy the first element to B, assuming it is the smallest

  MVI C,04 // Set counter C to the number of remaining elements (N-1 if array has N elements)

// Loop through the array to find the smallest number

FIND\_SMALLEST:   INX H // Move to the next element in the array

  MOV A,M // Load the current element into A

  CMP B // Compare the current element with the smallest found so far (in B)

  JC NEXT // If A >= B, skip to the next element

  MOV B,A // If A < B, update B with the new smallest value

NEXT:   DCR C // Decrement the counter

  JNZ FIND\_SMALLEST // If counter is not zero, repeat the loop

// Store the smallest number in memory location 2060H

  LXI H,2060 // Load HL with the address 2060H

  MOV M,B // Store the smallest number found (in B) into 2060H

  HLT // Halt the program

1. **Smallest Number**

; Load data into memory locations 2050H-2054H

LXI H, 2050H       ; Load HL with the start address 2050H

MVI A, 12H         ; Load A with the value 12H

MOV M, A           ; Store 12H at 2050H

INX H              ; Increment HL to point to 2051H

MVI A, 34H         ; Load A with the value 34H

MOV M, A           ; Store 34H at 2051H

INX H              ; Increment HL to point to 2052H

MVI A, 56H         ; Load A with the value 56H

MOV M, A           ; Store 56H at 2052H

INX H              ; Increment HL to point to 2053H

MVI A, 78H         ; Load A with the value 78H

MOV M, A           ; Store 78H at 2053H

INX H              ; Increment HL to point to 2054H

MVI A, 9AH         ; Load A with the value 9AH

MOV M, A           ; Store 9AH at 2054H

; Finding the largest number in 2050H - 2054H

LXI H, 2050H       ; Load HL with the start address of the array (2050H)

MOV A, M           ; Load the first element into A

MOV B, A           ; Copy the first element to B, assuming it is the largest

MVI C, 04H         ; Set counter C to the number of remaining elements (N-1 if array has N elements)

; Loop through the array to find the largest number

FIND\_LARGEST:

    INX H              ; Move to the next element in the array

    MOV A, M           ; Load the current element into A

    CMP B              ; Compare the current element with the largest found so far (in B)

    JNC NEXT           ; If A <= B, skip to the next element

    MOV B, A           ; If A > B, update B with the new largest value

NEXT:

    DCR C              ; Decrement the counter

    JNZ FIND\_LARGEST   ; If counter is not zero, repeat the loop

; Store the largest number in memory location 2060H

LXI H, 2060H       ; Load HL with the address 2060H

MOV M, B           ; Store the largest number found (in B) into 2060H

HLT                ; Halt the program

1. **Sqaure**

// Program to find the square of a number

  MVI A,05h // Load A with the value 12H (the number to be squared)

  STA 2050 // Store the number in memory location 2050H

  LDA 2050 // Load the number to square into A

  MOV B,A // Copy the number into B, for successive addition

  MVI C,00 // Clear register C (to hold the result)

  MOV D,A // Copy the original number into D (used as a counter)

// Start multiplication by successive addition

SQUARE\_LOOP:   MOV A,C // Load A with the current result

  ADD B // Add the number (B) to the result (C)

  MOV C,A // Move the result back into C

  DCR D // Decrement the counter D

  JNZ SQUARE\_LOOP // Repeat the loop until D reaches zero

// Store the result (the square) in memory location 2060H

  STA 2060 // Store the square result at memory location 2060H

  HLT // Halt the program