

## Practical 1: Memory Operations

### *a. Store the data byte 32H into a given memory location*

#### **Program:**

```
assembly
Copy code
MVI A, 32H      ; Load 32H into accumulator
STA 2500H       ; Store the accumulator content into memory location 2500H
HLT             ; End of program
```

#### **Binary Calculation:**

- 32H in binary = 0011 0010
- This byte will be stored at location 2500H.

 **Final Result:** Memory location 2500H contains binary 0011 0010 (32H).

---

### *b. Exchange the contents of memory locations 2000H and 4000H*

#### **Program:**

```
assembly
Copy code
LDA 2000H       ; Load data from 2000H into A
MOV B, A        ; Move it to register B
LDA 4000H       ; Load data from 4000H into A
STA 2000H       ; Store A into 2000H
MOV A, B        ; Move B back to A
STA 4000H       ; Store A into 4000H
HLT             ; End of program
```

#### **Example Binary Calculation:**

Suppose:

- Memory 2000H = 5AH = 0101 1010
- Memory 4000H = 3CH = 0011 1100

Exchange Process:

- 2000H becomes 3CH (0011 1100)
- 4000H becomes 5AH (0101 1010)

### ✅ Final Result:

- 2000H = <sub>3C</sub> (0011 1100)
  - 4000H = <sub>5A</sub> (0101 1010)
- 

Shall I continue with **Practical 2 (Addition & Subtraction)** next? I'll also show 8-bit and 16-bit binary calculations for each.

4o

You said:

yes

*ChatGPT said:*

Great! Let's move ahead with **Practical 2: Simple Assembly Language Programs**, with Assembly code, binary calculations, and final results. 🧠💻

---

## 📊 Practical 2: Simple Arithmetic Programs

---

### a. Addition of two 8-bit numbers

#### Program:

```
assembly
Copy code
MVI A, 25H      ; Load first number (25H) into accumulator A
MVI B, 18H      ; Load second number (18H) into register B
ADD B           ; Add B to A (A = A + B)
STA 2500H       ; Store result at memory 2500H
HLT
```

#### Binary Calculation:

- 25H = 0010 0101
- 18H = 0001 1000
- Binary addition:

yaml

```

Copy code
  0010 0101
+ 0001 1000
-----
  0011 1101 = 3D

```

✅ **Final Result:** Memory 2500H = 3D (Binary: 0011 1101)

---

## b. Subtraction of two 8-bit numbers

### Program:

```

assembly
Copy code
MVI A, 40H      ; Load first number (40H) into A
MVI B, 25H      ; Load second number (25H) into B
SUB B           ; A = A - B
STA 2501H       ; Store result at memory 2501H
HLT

```

### Binary Calculation:

- 40H = 0100 0000
- 25H = 0010 0101
- Binary subtraction:

```

yaml
Copy code
  0100 0000
- 0010 0101
-----
  0001 1011 = 1B

```

✅ **Final Result:** Memory 2501H = 1B (Binary: 0001 1011)

---

## c. Addition of two 16-bit numbers

### Assume:

- First number: 1234H
- Second number: 4321H

### Program:

```

assembly
Copy code
LXI H, 1234H    ; Load first 16-bit number
LXI D, 4321H    ; Load second 16-bit number
DAD D           ; HL = HL + DE
SHLD 2502H      ; Store result at memory 2502H and 2503H
HLT

```

### Binary Calculation:

- 1234H = 0001 0010 0011 0100
- 4321H = 0100 0011 0010 0001
- Add:

```

yaml
Copy code
0001 0010 0011 0100

```


- 0100 0011 0010 0001

0101 0101 0101 0101 = 5555H

```

yaml
Copy code

```

 \*\*Final Result:\*\* Memory 2502H = 55H, 2503H = 55H (5555H)

---

### \*\*d. Subtraction of two 16-bit numbers\*\*

Assume:

- First number: 5000H
- Second number: 1234H

**\*\*Program (8085 doesn't directly support 16-bit subtraction, so we do it manually):\*\***

```

````assembly
LXI H, 5000H    ; First number
LXI D, 1234H    ; Second number
MOV A, E        ; Low byte subtraction
SUB L
MOV L, A
MOV A, D        ; High byte subtraction
SBB H
MOV H, A
SHLD 2504H      ; Store result
HLT

```

### Binary Calculation:

- 5000H = 0101 0000 0000 0000
- 1234H = 0001 0010 0011 0100
- Subtract:

```
yaml
Copy code
0101 0000 0000 0000
```

- 0001 0010 0011 0100

0100 0001 1100 1100 = 3ECC (Hex)

```
yaml
Copy code
```

✅ \*\*Final Result:\*\* Memory 2504H = CCH, 2505H = 3EH → Final: 3ECCH



---

Ready for **\*\*Practical 3: BCD Packing & Masking\*\***? Or want a pause to review this part first?

You said:

continue

*ChatGPT said:*

Perfect! Let's move forward with **Practical 3: Packing and Unpacking Operations**  , covering Assembly programs, binary explanation, and final output.

## Practical 3: Packing and Masking Operations

a. Pack the two unpacked BCD numbers stored in memory locations

**Assumption:**

- Memory 2500H = 09H (BCD for 09)
- Memory 2501H = 08H (BCD for 08)
- Packed BCD = 98H (upper nibble = 9, lower = 8)

**Program:**


```

assembly
Copy code
LDA 2500H      ; Load 09H (MSB)
MOV B, A
LDA 2501H      ; Load 08H (LSB)
ANI 0FH        ; Mask upper nibble (keep lower)
MOV C, A
MOV A, B
RLC            ; Shift MSB left by 4 bits (×16)
RLC
RLC
RLC
ANI F0H        ; Mask lower nibble
ADD C          ; Add LSB
STA 2502H      ; Store packed result
HLT

```

### Binary Calculation:

- 09H = 0000 1001
- 08H = 0000 1000
- Packing:
  - Shift 09H left 4 bits → 1001 0000
  - OR with 0000 1000 → 1001 1000 = 98H

 **Final Result:** Memory 2502H = 98H (Binary: 1001 1000)

---

### b. Mask lower nibble of given data

#### Assume:

- Data at 2503H = 5CH (0101 1100)

#### Program:

```

assembly
Copy code
LDA 2503H
ANI F0H        ; Mask lower nibble (keep upper)
STA 2504H
HLT

```

### Binary Calculation:

- 5CH = 0101 1100
- AND with F0H = 1111 0000

yaml  
Copy code  
0101 1100

## & 1111 0000

0101 0000 = 50H

yaml  
Copy code

✓ \*\*Final Result:\*\* Memory 2504H = `50H` (Binary: `0101 0000`)

---

### \*\*c. Mask upper nibble of given data\*\*

\*\*Assume:\*\*

- Data at 2505H = `3AH` (0011 1010)

\*\*Program:\*\*

```assembly

LDA 2505H

ANI 0FH ; Mask upper nibble

STA 2506H

HLT

### Binary Calculation:

- 3AH = 0011 1010
- AND with 0FH = 0000 1111

yaml  
Copy code  
0011 1010

## & 0000 1111

0000 1010 = 0AH

yaml  
Copy code

✓ \*\*Final Result:\*\* Memory 2506H = `0AH` (Binary: `0000 1010`)

---

Next up: \*\*Practical 4 - Register Operations\*\* 📄 (shift, complement, etc.).  
Shall I continue?

Awesome! Let's now work on **Practical 4: Register Operations** 🧠🔄 — we'll cover shifting (right, left) and 1's complement with Assembly code, binary working, and results.

---

## 🔗 Practical 4: Register Operations

---

### a. Shift an 8-bit data four bits right (Data in register C)

**Assume:** C = B2H (Binary: 1011 0010)

#### Program:

```
assembly
Copy code
MVI C, B2H      ; Load 8-bit data into register C
MOV A, C        ; Move C to A for operations
RAR             ; Shift right 1 bit
RAR
RAR
RAR
MOV C, A        ; Store result back in C
HLT
```

#### Binary Calculation:

- Initial: 1011 0010
- After 4 right shifts:
  - 1st: 0101 1001
  - 2nd: 0010 1100
  - 3rd: 0001 0110
  - 4th: 0000 1011 = 0BH

✅ **Final Result:** Register C = 0B (Binary: 0000 1011)

---

### b. Shift a 16-bit data 1 bit left (Data in HL register pair)

**Assume:** HL = 1234H = 0001 0010 0011 0100




### Program:

```
assembly
Copy code
LXI H, 1234H      ; Load 16-bit number
MOV A, L          ; Lower byte in A
RAL               ; Rotate A left (carry goes to CY)
MOV L, A
MOV A, H          ; Higher byte in A
RAL               ; Rotate A left with carry
MOV H, A
HLT
```

### Binary Calculation:

- HL = 1234H = 0001 0010 0011 0100
- After 1-bit left shift:
  - New: 0010 0100 0110 1000 = 2468H

 **Final Result:** HL = 2468H (Binary: 0010 0100 0110 1000)

---

### c. Find One's complement of number stored in register


**Assume:** A = 3CH (Binary: 0011 1100)

### Program:

```
assembly
Copy code
MVI A, 3CH        ; Load number
CMA               ; Complement A
HLT
```

### Binary Calculation:

- 3C = 0011 1100
- 1's complement = 1100 0011 = C3H

 **Final Result:** A = C3H (Binary: 1100 0011)

---

#### d. Find One's complement of number in a memory location

**Assume:** Memory 2500H = 6AH (0110 1010)

**Program:**

```
assembly
Copy code
LDA 2500H
CMA
STA 2501H
HLT
```

**Binary Calculation:**

- 6AH = 0110 1010
- 1's complement = 1001 0101 = 95H

✅ **Final Result:** Memory 2501H = 95H (Binary: 1001 0101)

---

### Practical 5: Multiple Memory Operations

---

#### a. Add 16-bit numbers at 4000H & 4001H to those at 4002H & 4003H

**Assume:**

- 4000H = 34H (Low Byte), 4001H = 12H → Number 1 = 1234H
- 4002H = 78H (Low Byte), 4003H = 56H → Number 2 = 5678H

**Program:**

```
assembly
Copy code
LDA 4000H      ; Load low byte of first number
MOV L, A
LDA 4001H      ; Load high byte of first number
MOV H, A       ; HL = 1234H

LDA 4002H      ; Load low byte of second number
MOV E, A
LDA 4003H      ; Load high byte of second number
MOV D, A       ; DE = 5678H
```

```
DAD D          ; HL = HL + DE

SHLD 4004H     ; Store result at 4004H (low) and 4005H (high)
HLT
```

## Binary Calculation:

- 1234H = 0001 0010 0011 0100
- 5678H = 0101 0110 0111 1000
- Add:

```
yaml
Copy code
0001 0010 0011 0100
```

- 0101 0110 0111 1000

0110 1000 1010 1100 = 68ACH

markdown  
Copy code



**\*\*Final Result:\*\***  
- 4004H = `ACH`, 4005H = `68H` → Stored: `68ACH`

---

### **\*\*b. Subtract 16-bit number at 4002H & 4003H from 4000H & 4001H\*\***

**\*\*Using same values as above:\*\***  
- Number 1 (HL) = 1234H  
- Number 2 (DE) = 5678H

```
**Program:**
```assembly
LDA 4000H      ; Low byte
MOV L, A
LDA 4001H
MOV H, A       ; HL = 1234H

LDA 4002H
MOV E, A
LDA 4003H
MOV D, A       ; DE = 5678H

MOV A, L
SUB E
MOV L, A
```

```

MOV A, H
SBB D
MOV H, A

SHLD 4006H      ; Store result at 4006H (low) and 4007H (high)
HLT

```

### Binary Calculation:

- 1234H = 0001 0010 0011 0100
- 5678H = 0101 0110 0111 1000
- Subtraction:

```

Copy code
0001 0010 0011 0100

```

- 0101 0110 0111 1000

---

1011 1100 1011 0100 = BCB4H (2's complement format, negative)

```

yaml
Copy code

```

✅ **\*\*Final Result:\*\***  
 - 4006H = B4H, 4007H = BCH → Result: `BCB4H` (signed)

---

Ready to go ahead with **\*\*Practical 6: Calculations with respect to memory locations\*\***, including 2's complement and 1's complement from memory?

---

## Practical 6: Calculations on Memory

---

a. Add contents of 4000H and 4001H and place in 4002H & 4003H


Assume:

- 4000H = 78H
- 4001H = 56H → Together = 5678H

Let's add both as a 16-bit number.

**Program:**

```
assembly
Copy code
LDA 4000H
MOV L, A
LDA 4001H
MOV H, A      ; HL = 5678H
SHLD 4002H    ; Store into 4002H and 4003H
HLT
```

 **Result:** 4002H = 78H, 4003H = 56H → 5678H

---

**b. Subtract 4001H from 2000H and store in 4002H**

Assume:


- 2000H = A0H
- 4001H = 34H

**Program:**

```
assembly
Copy code
LDA 2000H
MOV B, A
LDA 4001H
MOV C, A
MOV A, B
SUB C
STA 4002H
HLT
```

**Binary:**

- A0H = 1010 0000, 34H = 0011 0100
- A0 - 34 = 0110 1100 = 6CH

 **Result:** 4002H = 6CH

---

### c. One's complement of number in memory

Assume:


- 2500H = 33H

#### Program:

```
assembly
Copy code
LDA 2500H
CMA
STA 2501H
HLT
```

#### Binary:

- 0011 0011 → One's comp = 1100 1100 = CCH

 **Result:** 2501H = CCH

---


### d. Two's complement of number in memory

#### Program:

```
assembly
Copy code
LDA 2500H
CMA
ADI 01H
STA 2502H
HLT
```

**If original = 33H:**

- One's comp = CCH
- +1 = CDH

 **Result:** 2502H = CDH

---

## Practical 7: String Operations

---

### a. Sum of a series of 5 numbers stored from 3000H

**Assume:** [3000H to 3004H] = 10H, 20H, 30H, 40H, 50H

**Program:**


```
assembly
Copy code
LXI H, 3000H
MVI C, 05H      ; Counter
MVI A, 00H      ; Accumulator clear

LOOP: ADD M
INX H
DCR C
JNZ LOOP

STA 3005H      ; Store sum
HLT
```

**Binary Sum:**

- $10 + 20 + 30 + 40 + 50 = \text{E0H} = 1110\ 0000$

 **Result:** 3005H = E0H

---

## Practical 8: BCD Operations

---

### a. Add 2 eight-bit BCD numbers

Assume:

- A = 25H
- B = 37H

**Program:**

```
assembly
Copy code
```

```
MVI A, 25H
ADI 37H
DAA
STA 3100H
HLT
```

### Binary Addition:

- $25 + 37 = 5CH$
- After DAA  $\rightarrow 62H$  (Valid BCD)

✅ **Result:** 3100H = 62H

---

### b. Add 4-digit BCD in HL & DE; store in 2300H & 2301H

Assume:

- HL = 1234H, DE = 4321H

### Program:

```
assembly
Copy code
LXI H, 1234H
LXI D, 4321H
DAD D
SHLD 2300H
HLT
```

### Addition:

- $1234 + 4321 = 5555H$

✅ **Result:** 2300H = 55H, 2301H = 55H

---

## Practical 9: Transfer 10 bytes from 2200H to 2300H

### Program:

```
assembly
Copy code
LXI H, 2200H      ; Source
```



```
LXI D, 2300H      ; Destination
MVI C, 0AH        ; Counter = 10
```

```
LOOP: MOV A, M
STAX D
INX H
INX D
DCR C
JNZ LOOP
HLT
```

**Result:** Data copied byte-by-byte from 2200H–2209H → 2300H–2309H