Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report
on
# “Even Number Separation”

[Code No:COMP 231]

Submitted by:

Abiral Adhikari (02)

Aakriti Banjara (05)

Nischal Baral (06)

Prashant Manandhar(30)

Samir Wagle (60)

Submitted to:

Dr. Gajendra Sharma
Department of Computer Science and Engineering

Submission Date: 15/05/2023

## Problem/Task

Write an assembly language program to separate even numbers from the given list of 50 numbers and store them in another list starting from 2200H. Assume the starting address of the 50 number list is 2100H.

## Instructions Used

- LXI - Load immediate data to register a pair. It loads a register pair with a 16-bit instantaneous value. Using the instruction LXI H, 2100H, the value 2100H is loaded into the H and L registers.

- MVI - Move immediate data to register. It loads an 8-bit immediate value to a register. For example, MVI B, 50 loads the value 50 to the B register.

- MOV - Move data between registers/memory locations. It transfers data between registers or memory locations. In the case of MOV A, M, the information from the memory location indicated to by the HL register pair is transferred to the accumulator register.

- ADI - Add immediate data to accumulator.The accumulator register gains an 8-bit instantaneous value as a result. For instance, ADI 7H increases the accumulator's value by 7H.

- INX - Increment register pair.  A 16-bit register pair is increased. For instance, the HL register pair is increased by INX H.

- DCR - Decrement register/memory location.An 8-bit register or memory location is decreased. DCR B, for instance, decreases the B register.

- JNZ - Jump if not zero.If the zero flag is not set, it leaps to a specific point in the memory. For instance, if the zero flag is not set, JNZ loop jumps to the memory location designated as "loop."

- ANI - Logical AND immediate data with accumulator. It does a logical AND operation between the accumulator register and an 8-bit instantaneous value. For instance, ANI

01H executes a logical AND operation between the value 01H and the accumulator register.

- STAX - Store accumulator data to memory.The memory address pointed to by the register pair receives the contents of the accumulator register. As an illustration, the DE register pair points to the memory address where STAX D holds the contents of the accumulator register.

- HLT - Halt the CPU.  It halts the running of the program. HLT, for instance, halts the running of the program.

# Flowchart

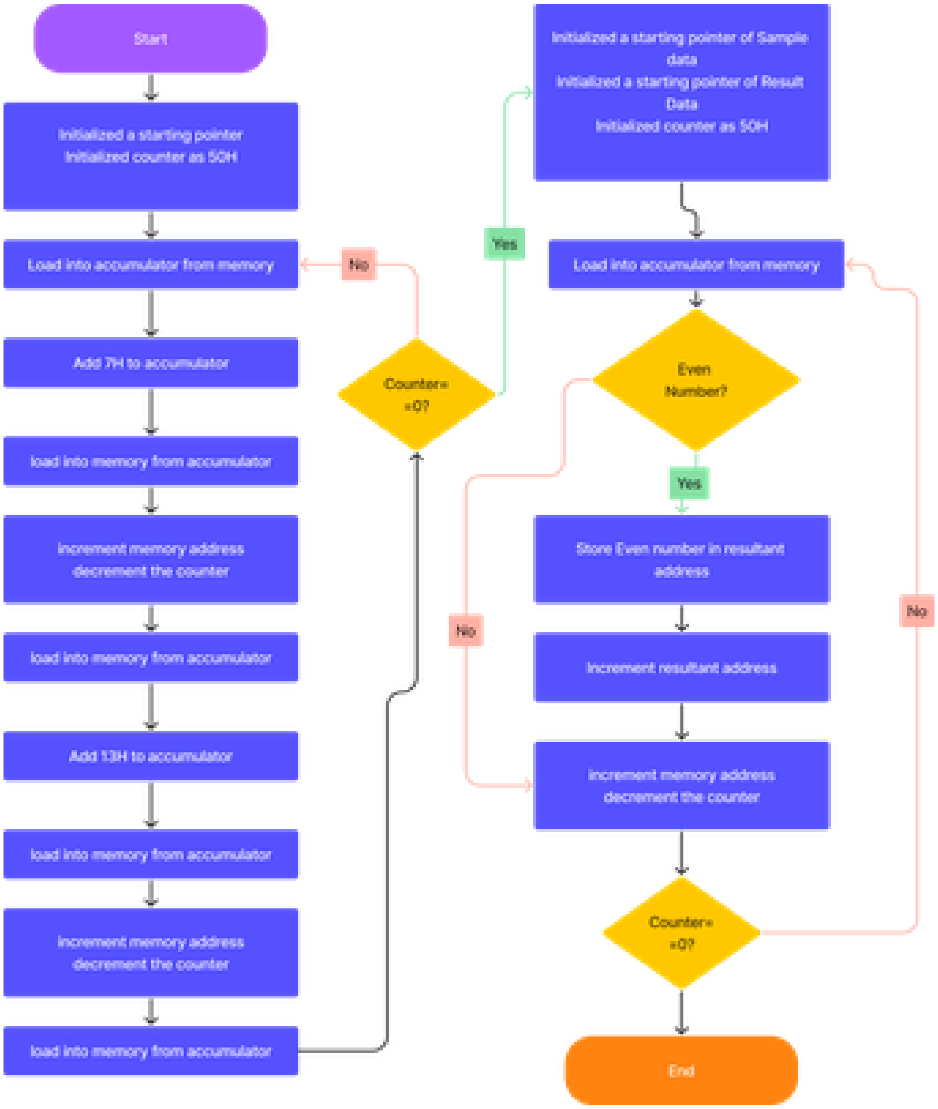

Start
Initialized a starting pointer
Initialized counter as 50H
Load into accumulator from memory
Add 7H to accumulator
load into memory from accumulator
increment memory address
decrement the counter
load into memory from accumulator
Add 13H to accumulator
load into memory from accumulator
increment memory address
decrement the counter
load into memory from accumulator
Counter=
=0?
No
Yes
Initialized a starting pointer of Sample data
Initialized a starting pointer of Result Data
Initialized counter as 50H
Load into accumulator from memory
Even
Number?
Yes
No
Store Even number in resultant address
Increment resultant address
increment memory address
decrement the counter
Counter=
=0?
No
End

*Figure 1: Flowchart*

# Procedure/Methods

- The starting memory address where the data is placed is used by the program to initialize the H register. It is set to 2100H in this instance.
- The B register, which serves as a counter for the loop, is initialized by the program with a value of 50.
- The C register is initialized by the program with the value 7H (7 in hexadecimal).
- The program then starts a loop that uses the H register as the memory address to obtain a number from memory.
- The software uses the accumulator register to add the value 7H to the retrieved number before storing the outcome back in memory.
- The next number is read from memory by the program, which then uses the accumulator register to add the value 13H (or 13 in hexadecimal) to it before writing the result back to memory.
- The H register is increased by the program to point to the following memory location.
- The B register counter is decreased by one by the program.
- The B register counter is verified to be zero by the software. If not, the loop is completed by returning to step 4.
- The program terminates if the B register counter is 0.
- The starting memory address where the data is placed is used by the program to initialize the H register. It is set to 2100H in this instance.
- The starting memory address where the result data will be stored is used to initialize the D register by the application. It is set to 2200H in this instance.
- The C register is filled with the value 50H, or 80 in decimal, by the program.
- The program then starts a loop that uses the H register as the memory address to obtain a number from memory.
- Using the AND operator and the mask value of 01H, the software determines whether the retrieved number is even present.
- If the recovered number is even, the software uses the STAX instruction to store it in the memory address indicated to by the D register.
- To point to the subsequent result memory address, the software advances the D register.
- The H register is increased by the program to point to the following memory location.
- The C register counter is decreased by one by the program.
- The C register counter is verified to be zero by the program. If not, the loop is completed by returning to step 4.
- The program terminates if the C register counter is 0.

## Source Code

| Mnemonic | | Comments |
|---|---|---|
| | LXI H, 2100H | Initialize H register with starting memory address |
| | MVI B, 32H | Set the counter to 50 |
| | MVI C, 7H | Loads the value 7H (7 in hexadecimal) into the register C |
| | | |
| Loop: | MOV A, M | Move the number at current memory location to the accumulator |
| | ADI 7H | Adds the value 7H (7 in hexadecimal) to the accumulator register |
| | MOV M, A | Move the updated number back to current memory location |
| | INX H | Increment memory address |
| | DCR B | Decrement the counter |
| | MOV M, A | Moves the contents of the accumulator register to the memory location pointed by the HL register pair |
| | MOV A, M | Move the number at current memory location to accumulator |
| | ADI 13H | Adds the value 13H (13 in hexadecimal) to the accumulator register |
| | MOV M, A | Move the updated number back to current memory location |

| | | |
|---|---|---|
| | INX H | Increment memory address |
| | DCR B | Decrement the counter |
| | MOV M, A | Moves the contents of the accumulator register to the memory location pointed by the HL register pair |
| | JNZ loop | Jump back to loop until the counter becomes 0 |
| | | |
| | LXI H, 2100H | Initialize H register with starting memory address |
| | LXI D, 2200H | Initialize memory pointer 2 |
| | MVI C, 50H | Initialize counter |
| | | |
| BACK: | MOV A, M | Get the number |
| | ANI 01H | Check for even number |
| | JNZ SKIP | If ODD, don't store |
| | MOV A, M | Get the number |
| | STAX D | Store the number in result list |
| | INX D | Increment pointer 2 |
| | | |
| SKIP | INX H | Increment pointer l |
| | DCR C | Decrement counter |
| | JNZ BACK | If not zero, repeat |
| | | |
| | HLT | Stop |

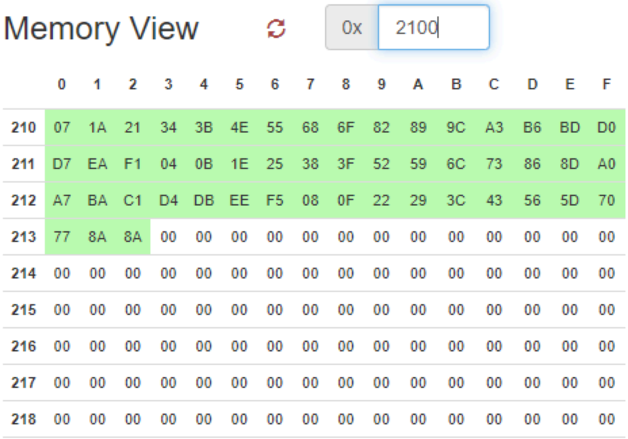
## Assembler Output

| Code | Mnemonic | Comments |
|---|---|---|
| 21 00 21 | LXI H, 2100H | Initialize H register with starting memory address |
| 06 32 | MVI B, 32H | Set the counter to 50 |
| 0E 07 | MVI C, 7H | Loads the value 7H (7 in hexadecimal) into the register C |
| 7E | **loop:** MOV A, M | Move the number at current memory location to the accumulator |
| C6 07 | ADI 7H | Adds the value 7H (7 in hexadecimal) to the accumulator register |
| 77 | MOV M, A | Move the updated number back to current memory location |
| 23 | INX H | Increment memory address |
| 05 | DCR B | Decrement the counter |
| 77 | MOV M, A | Moves the contents of the accumulator register to the memory location pointed by the HL register pair |
| 7E | MOV A, M | Move the number at current memory location to accumulator |
| C6 13 | ADI 13H | Adds the value 13H (13 in hexadecimal) to the accumulator register |
| 77 | MOV M, A | Move the updated number back to current memory location |
| 23 | INX H | Increment memory address |
| 05 | DCR B | Decrement the counter |
| 77 | MOV M, A | Moves the contents of the accumulator register to the memory location pointed by the HL register pair |
| C2 07 08 | JNZ loop | Jump back to loop until the counter becomes 0 |
| 21 00 21 | LXI H, 2100H | Initialize H register with starting memory address |
| 11 00 22 | LXI D, 2200H | Initialize memory pointer 2 |
| 0E 50 | MVI C, 50H | Initialize counter |
| 7E | **BACK**: MOV A, M | Get the number |

| | | |
|---|---|---|
| E6 01 | ANI 01H | Check for even number |
| C2 29 08 | JNZ SKIP | If ODD, don't store |
| 7E | MOV A, M | Get the number |
| 12 | STAX D | Store the number in result list |
| 13 | INX D | Increment pointer 2 |
| 23 | **SKIP:** INX H | Increment pointer l |
| 0D | DCR C | Decrement counter |
| C2 20 08 | JNZ BACK | If not zero, repeat |
| 76 | HLT | Stop |

# Output

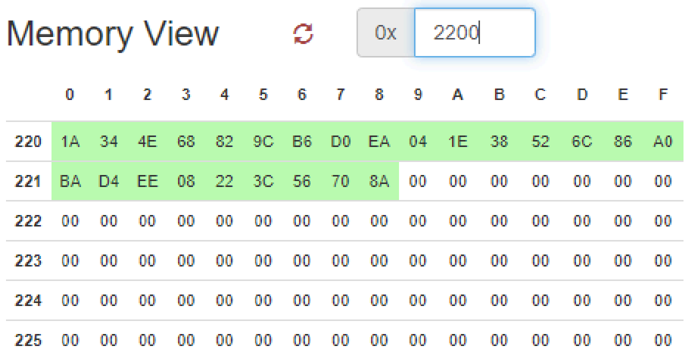
When the address is 2100H for input of 50 number needed for the arranging . which is shown below:



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 210 | 07 | 1A | 21 | 34 | 3B | 4E | 55 | 68 | 6F | 82 | 89 | 9C | A3 | B6 | BD | D0 |
| 211 | D7 | EA | F1 | 04 | 0B | 1E | 25 | 38 | 3F | 52 | 59 | 6C | 73 | 86 | 8D | A0 |
| 212 | A7 | BA | C1 | D4 | DB | EE | F5 | 08 | 0F | 22 | 29 | 3C | 43 | 56 | 5D | 70 |
| 213 | 77 | 8A | 8A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 214 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 215 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 216 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 217 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 218 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

*Figure 1: Memory block for showing 50 number input*

The list after filtering out the even number is shown in figure below which was stored in address 2200H:

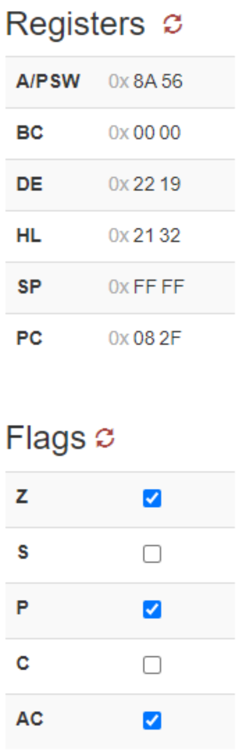


Memory View 0x 2200

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 220 | 1A | 34 | 4E | 68 | 82 | 9C | B6 | D0 | EA | 04 | 1E | 38 | 52 | 6C | 86 | A0 |
| 221 | BA | D4 | EE | 08 | 22 | 3C | 56 | 70 | 8A | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 222 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 223 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 224 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 225 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

*Figure 3: Memory block showing filtered out numbers*

The flag and register output are shown below:



Registers

| | |
|---|---|
| A/PSW | 0x 8A 56 |
| BC | 0x 00 00 |
| DE | 0x 22 19 |
| HL | 0x 21 32 |
| SP | 0x FF FF |
| PC | 0x 08 2F |

Flags

| | |
|---|---|
| Z | ☑ |
| S | □ |
| P | ☑ |
| C | □ |
| AC | ☑ |

*Figure 4: Register and Flag of status*