Lab 2: Addressing Mode

- **Link**
  - Document
  - Video

- **Basic** (50%)
  - **Description**

    Store two unsigned 8-bit numbers in memory at addresses [0x120] and [0x121], respectively. Use **indirect addressing** to access memory locations from 0x120 to 0x126. Next, compute the values for memory addresses [0x122] through [0x126] based on the values stored at the two previous addresses:

    1. If the target address is even (e.g., [0x122]), set its value to the sum of the two preceding addresses: [addr - 2] + [addr - 1].
    2. If the target address is odd (e.g., [0x123]), set its value to the difference: [addr - 2] - [addr - 1].

    Note: All operations must use 8-bit arithmetic with wraparound behavior (i.e., modulo 256).

  - **Example**

Address	Testcase 1	Testcase 2
[0x120]	0x02	0xFE
[0x121]	0x03	0xEE
[0x122]	0x05	0xEC
[0x123]	0xFE	0x02
[0x124]	0x03	0xEE
[0x125]	0xFB	0x14
[0x126]	0xFE	0x02

  - **Standard of grading**
    1. All read and write operations on memory locations [0x120] through [0x126] must be performed by **using indirect addressing**.
    2. All computed results must be stored in the **correct and specified memory locations**.

- **Advance** (30%)
  - **Description**

    Store two **sorted sequences** A and B in ascending order. Sequence A contains six unsigned numbers at addresses [0x200] through [0x205]. Sequence B contains five unsigned numbers at addresses [0x210] through [0x214]. Please use **indirect addressing** to access the memory locations of sequences A and B. Then, merge sequence A and B into a single **ascending** sequence, and store the results at the addresses [0x220] through [0x22A].

    Note: If two values are equal, the value from Sequence A should come first. This ensures a **stable merge**.

  - **Example**

    Testcase1:

    Sequence A: [0x01, 0x03, 0x05, 0x07, 0x09, 0x0B]

    Sequence B: [0x02, 0x04, 0x06, 0x08, 0x0A]

    Testcase 2:

    Sequence A: [0x05, 0x1A, 0x2C, 0x40, 0x7E, 0xA0]

    Sequence B: [0x06, 0x20, 0x33, 0x80, 0xF0]

Address	Testcase 1	Testcase 2
[0x220]	0x01	0x05
[0x221]	0x02	0x06
[0x222]	0x03	0x1A
[0x223]	0x04	0x20
[0x224]	0x05	0x2C
[0x225]	0x06	0x33
[0x226]	0x07	0x40
[0x227]	0x08	0x7E
[0x228]	0x09	0x80
[0x229]	0x0A	0xA0
[0x22A]	0x0B	0xF0

  - **Standard of grading**

    1. All read and write operations on the memory locations of Sequence A and Sequence B must performed by **using indirect addressing**.
    2. Sequence A, Sequence B and all computed results must be stored in the **correct and specified memory locations**, with the merging performed using a **stable merge**.

- **Hard** (20%)
  - **Description**

    Given 8 unsigned bytes, each byte is treated as a pair of nibbles (4-bit values): a high nibble and a low nibble. Determine whether there exists a rearrangement at the byte level such that the entire sequence of nibbles forms a **palindrome** (i.e., it reads the same forwards and backwards).

    If such a rearrangement exists:

    Store the rearranged 8-byte sequence at addresses [0x320] through [0x327].

    Among each mirrored nibble pair, the smaller value should appear in the first half ([0x320] to [0x323]) in **ascending** order.

    If no such palindrome can be formed:

    Store 0xFF in all addresses from [0x320] through [0x327].

    Note: The input will never consist of all 0xFF bytes.

  - **Example**

    Testcase1:

    [0x63, 0x2B, 0xAA, 0xFD, 0xAA, 0xB2, 0x36, 0xDF]

    Testcase 2:

    [0x63, 0x2B, 0xAA, 0xFD, 0xAB, 0xB2, 0x36, 0xDF]

Address	Testcase 1	Testcase 2
[0x320]	0x2B	0xFF
[0x321]	0x36	0xFF
[0x322]	0xAA	0xFF
[0x323]	0xDF	0xFF
[0x324]	0xFD	0xFF
[0x325]	0xAA	0xFF
[0x326]	0x63	0xFF
[0x327]	0xB2	0xFF

  - **Standard of grading**

    1. Use at least one type of **indirect addressing register**.
    2. All results must be stored in the **correct order and specified memory locations**.