Lab 4: Macro and Subroutine

- **Link**
  - Document
  - Video

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

    Your task is to develop a macro that performs the following sequence of operations:
    1. Accept two 16-bit unsigned integers as input:
       - The first number x is stored with its high byte at memory address 0x002 and low byte at 0x003.
       - The second number y is stored with its high byte at 0x004 and low byte at 0x005.
    2. Performs a bitwise AND operation on the two input numbers.
    3. Stores the 16-bit result of the AND operation as two separate bytes:
       - The high byte at memory address 0x000.
       - The low byte at memory address 0x001.
    4. Multiplies the two 8-bit values now stored at 0x000 and 0x001.
    5. Stores the final 16-bit product in memory:
       - The high byte at address 0x010.
       - The low byte at address 0x011.

  - **Example**
    - Testcase 1

Address	0x002	0x003	0x004	0x005
Value	0xAB	0xCD	0x56	0x78

Address	0x000	0x001	0x010	0x011
Value	0x02	0x48	0x00	0x90

  - **Standard of grading**
    1. The MACRO definition follows **And_Mul xh, xl, yh, yl**.
    2. All results must be stored in the correct and specified memory locations.

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

    Design a subroutine named division that performs unsigned 16-bit integer division. The inputs are:

- Dividend: high byte at 0x000, low byte at 0x001
- Divisor: high byte at 0x002, low byte at 0x003

Upon completing the division:
- The quotient must be stored with its high byte at 0x010 and low byte at 0x011
- The remainder must be stored with its high byte at 0x012 and low byte at 0x013

Assume that:
- The divisor is non-zero
- The quotient fits within 16 bits

■ **Example**

Address	Testcase 1	Testcase 2
[0x000]	0x00	0x01
[0x001]	0x64	0xF4
[0x002]	0x00	0x00
[0x003]	0x0A	0x22
[0x010]	0x00	0x00
[0x011]	0x0A	0x0E
[0x012]	0x00	0x00
[0x013]	0x00	0x18

■ **Standard of grading**
1. There must be a **subroutine named "division"**.
2. The results of the quotient and remainder must be stored in the correct memory addresses.

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

    Implement Newton's Method using **16-bit unsigned integer arithmetic** to compute the square root of a number. The iterative formula is:

$$x_{n+1} = \left\lfloor \frac{x_n + \left\lfloor \frac{N}{x_n} \right\rfloor}{2} \right\rfloor$$

Because the computation is restricted to unsigned 16-bit integers, the input values N and x_0 are both **16-bit unsigned integers**. The value N **must be a nonzero perfect square**, and the initial guess x_0 **must be at least 1**. Under these conditions, the sequence $\{x_0, x_1, \ldots, x_n\}$ will eventually reach the exact square root of N.

Please design a subroutine named "**newtonSqrt**" to implement Newton's method using the iteration formula given above. The value N is stored with its high byte at address [0x020] and low byte at [0x021], and x_0 is stored with its high byte at [0x022] and low byte at [0x023]. After completing the Newton iterations, the square root of N must be written with its high byte at [0x024] and low byte at [0x025].

You may use the stopping condition $x_{n+1} = x_n$ for the iteration, and you may also reuse the division subroutine from the previous question. All intermediate computations fit within 16-bit unsigned integers.

- **Example**

Address	Testcase 1	Testcase 2
[0x020]	0x00	0xFE
[0x021]	0x40	0x01
[0x022]	0x00	0x55
[0x023]	0x78	0x66
[0x024]	0x00	0x00
[0x025]	0x08	0xFF

- **Standard of grading**
  1. There must be a **subroutine named "newtonSqrt"**.
  2. The result must be stored in the correct memory addresses.
  3. Newton's Method must be applied using the provided iteration formula to compute the square root.

- Note: If you have any questions, please send an email to the TA mailbox (mprocessor2025@gmail.com). Emails sent to individual TAs will not be answered.