LAB 04 - Requirement Description

- Youtube 影片
  - Link: https://www.youtube.com/watch?v=MsbWbfyfJ2Y
- Hackmd 連結
  - Link: https://hackmd.io/@CM_CBTVtTuKNjJZ1yF2ptw/SJTv92-CA
- Lab requirements
  - **基本題(70%)**

➢ **題目敘述**

給定大小為 2 Bytes 的兩數 X 以及 Y，首先將兩數相減之結果 X-Y 存入[0x000], [0x001] 兩個記憶體位址中，將此相減結果的兩個 8-bit 結果各別視為一個數字，，將兩個 8-bit 相乘後的積再存入[0x010] 以及 [0x011] 兩個記憶體位址中，請根據上述要求請設計一 macro : Sub_Mul *xh*, xl, yh, yl 來完成題目要求。

➢ Macro 描述 (Sub_Mul xh, xl, yh, yl)

xh, xl 分別代表第一個數的 High byte 及 Low byte，yh, yl 亦然。請使用此 macro 將最終結果 High byte 存入[0x010], Low byte 存入[0x011]。

➢ **範例測資 (第一列為儲存位址)**

位置/變數名稱	xh	xl	yh	yl	[0x000]	[0x001]	[0x010]	[0x011]
範例 1	0x0A	0x04	0x04	0x02	0x06	0x02	0x00	0x0C
範例 2	0x02	0x0C	0x00	0x0F	0X01	0xFD	0x00	0xFD

➢ **評分標準**

1. 需使用 macro，且參數名稱需與上述一致
2. 答案需儲存於正確位置
3. x,y 儲存位置均不限制，可自行決定

● **進階題(30%)**

➢ **題目敘述:**

給定兩個三維向量(a = (a1, a2, a3), b = (b1 ,b2, b3))，並設計一個**名為 cross 的 subroutine**，來計算這兩個向量的外積 c = (c1, c2, c3)，並以二補數方式來呈現。

將答案 c (c1, c2, c3)儲存於([0x020], [0x021], [0x022])

(所有向量內值必定介於-128-127 之間)。

➢ **外積公式**

$$\vec{A}(a1, a2, a3) \times \vec{\mathrm{B}}(\mathrm{b1}, \mathrm{b2}, \mathrm{b3}) = (a2 * b3 - a3 * b2, a3 * b1 - a1 * b3, a1 * b2 - a2 * b1)$$

➢ **範例測資 (第一列為儲存位址):**

位置/變數名稱	a1	a2	a3	b1	b2	b3	[0x020]	[0x021]	[0x022]
範例 1	0x0B	0x00	0x10	0x0C	0x00	0x06	0x00	0x7E	0x00
範例 2	0x03	0x04	0x05	0x06	0x07	0x08	0xFD	0x06	0xFD

➢ **評分標準**

1. 需使用**名為 cross 的 subroutine**
2. 答案需儲存於正確位置

- **加分題(20%)**

➢ **題目敘述:**

設計一個**名為 fib 的 subroutine**，並輸入數字 n 來得到費氏數列的第 n 項(0 <= n <= 24)，最後將答案存於[0x000](high byte)及[0x001](low byte)。

➢ **費式數列公式:**

$F_0=0,$

$F_1=1,$

$F_{n+2} = F_n + F_{n+1}$

➢ **範例測資 (第一列為儲存位址):**

位置/變數名稱	n	[0x000]	[0x001]
範例 1	9	0x00	0x22
範例 2	15	0x02	0x62

➢ **評分標準**

1. 需使用**名為 fib 的 subroutine**。
2. 答案需儲存於正確位置
3. 不可針對測資寫死答案

LAB 04 - Requirement Description

- Youtube video
  - Link: https://www.youtube.com/watch?v=akHZKwJf8D4
- Hackmd
  - Link: https://www.youtube.com/watch?v=rDVW7ZjWRyg
- Lab requirements
  - Basic (70%)

➢ **Description:**

Given two 16-bit numbers, x and y, first subtract y from x, and store the result in memory locations [0x000] and [0x001], where the high byte is stored in [0x000] and the low byte in [0x001]. Then, treat the high and low bytes of the result as separate 8-bit numbers, multiply them, and store the product in memory locations [0x010] and [0x011], with the high byte of the product stored in [0x010] and the low byte in [0x011]. Please design a macro: Sub_Mul xh, xl, yh, yl to achieve this.

➢ **Macro description (Sub_Mul xh, xl, yh, yl):**

xh and xl represent the high byte and low byte of the first number, respectively, and yh and yl represent the same for the second number.
Please use this macro to store the final result's high byte in [0x010] and the low byte in [0x011].

➢ **Sample test data (First row is address):**

Address/variable	xh	xl	yh	yl	[0x000]	[0x001]	[0x010]	[0x011]
Example1	0x0A	0x04	0x04	0x02	0x06	0x02	0x00	0x0C
Example2	0x02	0x0C	0x00	0x0F	0X01	0xFD	0x00	0xFD

- **Criteria:**
  1. Please use the provided macro to complete this task, **ensuring that the macro name and parameter names match those mentioned above.**
  2. The result must be stored in correct locations.
  3. The storage locations of x and y are not restricted and can be decided freely.

- **Advanced (30%)**

Description:

Given two 3-dimensional vectors (a=(a1, a2, a3), b=(b1, b2, b3)), **design a subroutine called "cross"** to calculate the cross product of these two vectors, c = (c1, c2, c3), and present the result in two's complement format. Store the answers in memory locations [0x020] (c1), [0x021] (c2), and [0x022] (c3). All values in the vectors are guaranteed to be between -128 and 127.

cross product formula

$$\vec{A}(a1, a2, a3) \times \vec{B}(b1, b2, b3)$$
$$= (a2 * b3 - a3 * b2, a3 * b1 - a1 * b3, a1 * b2 - a2 * b1)$$

Sample test data:

Address/variable	a1	a2	a3	b1	b2	b3	[0x020]	[0x021]	[0x022]
Example1	0x0B	0x00	0x10	0x0C	0x00	0x06	0x00	0x7E	0x00
Example2	0x03	0x04	0x05	0x06	0x07	0x08	0xFD	0x06	0xFD

Criteria:

1. There must be a **subroutine named "cross"**.
2. The result must be stored in correct locations.

- Bonus (20%)

- **Description:**

  Design a **subroutine called "fib"**, which takes an input number n and calculates the n-th Fibonacci number (where 0 <= n <= 24). Finally, store the result in memory locations[0x000](high byte) and [0x001](low byte).

  The Fibonacci sequence is defined as:

  $F_0=0,$

  $F_1=1,$

  $F_{n+2} = F_n + F_{n+1}$

- **Sample test data:**

Address/variable	n	[0x000]	[0x001]
Example1	9	0x00	0x22
Example2	15	0x02	0x62

- **Criteria**

  1. There must be a subroutine named "fib".
  2. The result must be stored in correct locations.
  3. You must not hardcode the answers for the test cases.