LAB 04 - Requirement Description

- **連結**
  - **影片:**
    https://youtu.be/Q1G5joGdbMw
  - **Hackmd:**
    https://hackmd.io/@KueJ3OutS5W6uTv5MeC8dg/SkMnxbB6n

- **基本題 (70%) :**

➢ **題目敘述：**

給定兩數，將相加後得到的結果存入[0x000], [0x001]，將結果相乘後再存入[0x010], [0x011]，請設計一 macro : **Add_Mul xh, xl, yh, yl** 來達到要求。

➢ **Macro 描述 ：**

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 ：**

xh	xl	yh	yl	[0x000]	[0x001]	[0x010]	[0x011]
0x04	0x02	0x0A	0x04	0x0E	0x06	0x00	0x54
0x00	0xFF	0x02	0x0C	0x03	0x0B	0x00	0x21

➢ **評分標準 ：**

1. 請使用 macro 完成此題，命名和參數名稱需與上述一致。
2. 結果必須存放至[0x010], [0x011]中。

- **進階題 (30%) :**

➢ **題目敘述：**

給定兩個二維矩陣，對應的矩陣 A 元素分別為 a1, a2, a3, a4、矩陣 B 元素分別為 b1,b2,b3,b4。請撰寫一個名稱為 multiply 的 subroutine 來實作矩陣乘法(如圖所示)並將矩陣 C 元素分別的結果存入 [0x000](c1), [0x001](c2), [0x002](c3), [0x003](c4)。

$$\begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \begin{bmatrix} b1 & b2 \\ b3 & b4 \end{bmatrix} = \begin{bmatrix} c1 & c2 \\ c3 & c4 \end{bmatrix}$$

- **範例測資**：

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} c1 & c2 \\ c3 & c4 \end{bmatrix}$$

[0x000]	[0x001]	[0x002]	[0x003]
0x03	0x03	0x01	0x04

- **評分標準**：
  1. 測資所有矩陣內的元素都為非負整數，乘出的結果會限制在 8-Bits 範圍(0x00~0xFF)。
  2. 必須有個名稱為名稱為 multiply 的 subroutine
  3. 結果必須存放至[0x000], [0x001],[0x002],[0x003]中。

- **加分題 (20%)**：

- **題目敘述**：

請寫出一個名為 Fact 的 subroutine，實作以下組合公式：

$$C(n,k) = n!/k!\,(n-k)!$$

最後將答案存入[0x000]中。

(測資中 $0 \leq n \leq 10$, 且 $k < n$)

- **範例測資：**

n	k	[0x000]
5	2	0x0A
7	3	0x23
9	5	0x7E

- **評分標準：**
  1. 請使用名為 Fact 的 subroutine 完成此題。
  2. 必須使用指令 **RCALL**。
  3. 不可針對測資將程式邏輯寫死，助教會檢查。
  4. 結果必須存放至[0x000]。

提示:

1. **若直接計算階乘的結果可能會使用到太多 bit，建議與除法同時並行。**
2. **可以利用遞迴的方式（巴斯卡定理）計算組合公式。**

LAB 04 - Requirement Description

- **Link**
  - **Video:** https://youtu.be/Q1G5joGdbMw
  - **Hackmd:** https://hackmd.io/@KueJ3OutS5W6uTv5MeC8dg/SkMnxbB6n

- **Basic (70%) :**

➢ **Description** :

Given two numbers, add the results and store the sum in [0x000] and [0x001] (with the **high byte** stored in [0x000] and the **low byte** stored in [0x001]). Then, multiply the numbers stored in [0x000] and [0x001], then store the product in [0x010] and [0x011] (with the **high byte** stored in [0x010] and the **low byte** stored in [0x011]).

Please design a macro: **Add_Mul xh, xl, yh, yl** to achieve this.

➢ **Macro description** :

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** :

xh	xl	yh	yl	[0x000]	[0x001]	[0x010]	[0x011]
0x04	0x02	0x0A	0x04	0x0E	0x06	0x00	0x54
0x00	0xFF	0x02	0x0C	0x03	0x0B	0x00	0x21

➢ **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 [0x010] and [0x011].

● **Advanced (30%) :**

➢ **Description** :

Given two 2D matrices, the corresponding elements of matrix A are a1, a2, a3, a4, and the elements of matrix B are b1, b2, b3, b4. Please write a subroutine named "multiply" to implement matrix multiplication (as shown in the diagram) and store the results of matrix C elements into [0x000](c1), [0x001](c2), [0x002](c3), [0x003](c4) respectively.

$$\begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \begin{bmatrix} b1 & b2 \\ b3 & b4 \end{bmatrix} = \begin{bmatrix} c1 & c2 \\ c3 & c4 \end{bmatrix}$$

➢ **Sample test data** :

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} c1 & c2 \\ c3 & c4 \end{bmatrix}$$

[0x000]	[0x001]	[0x002]	[0x003]
0x03	0x03	0x01	0x04

➢ **Criteria** :

4. The test data for all elements within the matrices are non-negative integers. The multiplication result will be limited to the 8-Bits range (0x00~0xFF).
5. There must be a subroutine named "multiply".
6. The results must be stored in [0x000], [0x001], [0x002], [0x003] respectively.

● **Bonus (20%) :**

➢ **Description :**

Please write a subroutine named "Fact" to implement the following combination formula:

C(n,k) = n! / (k!(n-k)!)

Finally, store the answer in [0x000].

(Test data: 0 ≤ n ≤ 10 and k < n)

➢ **Sample test data :**

n	k	[0x000]
5	2	0x0A
7	3	0x23
9	5	0x7E

➢ **Criteria :**

1. Please use a subroutine named "Fact" to complete this task.
2. You must use the instructions **RCALL**.
3. The result must be stored in [0x000]

Hint:

1. **Directly calculating factorial results might require too many bits; consider parallel processing with division.**
2. **The combination formula can be calculated using a recursive approach (Pascal's Triangle).**