Lab 4: Requirement Description

10/9 (21:14)更新：更新了基礎題與進階題的內容。

- Macro & Subroutine 教學
  - 影片： https://youtu.be/XQF7c9myE1Y
  - Hackmd連結：https://hackmd.io/dlx_c18XSdWER4Y1gbkV2w

- Lab requirements:
  - **基本題 (70%)** ：
    - **題目敘述**：給出座標兩點A(x1,y1)、B(x2,y2)且 x1>x2 , y1>y2，設計以下macro 算出兩點間的距離平方：

      DIST x1, y1, x2, y2, F1, F2

    - **功能**：計算出兩點間的距離平方，前四個參數對應頂點座標x1,y1, x2,y2 的值，F1和F2則為答案存放的位置。

    - **舉例測資**：

      使用DIST 0x05, 0x07, 0x02, 0x03, 0x00, 0x01後：

      答案為(0x05 - 0x02)^2 + (0x07 - 0x03)^2 = **0x0019**。

      將答案的High Byte放入[0x000] , [0x000] = 0x00。

      將答案的Low Byte放入[0x001] , [0x001] = 0x19。

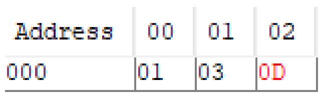
    - **評分標準**：
      1. 會檢查是否有建立並使用題目敘述的 macro，macro 的名稱和參數名稱需與敘述一致。
      2. 實體Demo測資與範例測資不同，請不要針對測資去將程式邏輯寫死。

  - **進階題 (30%)** ：
    - **題目敘述**：寫一個名為 GP 的 subroutine 算出等比數列的**前三項總和**，在 GP 裡需使用迴圈並以更改 program counter (PCL)取代 goto 以及 bra 指令，將結果放入位址 0x002中。

      首項和公比皆為正數，不會有負數的情況。

      答案總和會限制在8-Bits範圍 (0x00~0xFF)。

    - **舉例**：首項 = 1，公比 = 3 時，GP(3) = 1 + 3 + 9 = 13。



Address	00	01	02
000	01	03	0D

- **評分標準：**
  1. 會檢查是否有名為 GP 的 subroutine。
  2. 需使用到 rcall 指令。
  3. 需使用迴圈。
  4. 實體Demo測資與範例測資不同，請不要針對測資去將程式邏輯寫死。
  5. 在迴圈中必須以更改PCL的方式來取代goto或bra的使用。
  6. 不能出現 goto 以及 b開頭的branch 指令 (如 BRA、BZ、BN...)。
  7. 結果需放在位址 0x002。

- **加分題 (20%)：**
  - **題目敘述：** 寫一個名為Hanoitower 的 subroutine ，利用遞迴的方式計算出N個圓盤在河內塔上移動至另一桿所需的最少次數，將結果放入位址0x000當中。
  - **河內塔介紹：**https://zh.wikipedia.org/zh-tw/%E6%B1%89%E8%AF%BA%E5%A1%94
  - **舉例：**
    1. N=2時，答案為3，放入[0x000]。
    2. N=3時，答案為7，放入[0x000]。
    3. N=4時，答案為15，放入[0x000]。
  - **評分標準：**
    1. 會檢查是否有名為Hanoitower的soubroutine。
    2. 需用遞迴撰寫。
    3. 結果需存放在位址0x000。
  - **提示：**
    1. **不用顯示圓盤交換的過程**，僅需使用**遞迴**的方式求出**最少移動次數**即可。
    2. 可以用之前學過的**FSRx**來存放遞迴過程中的變數。

Lab 4: Requirement Description

10/9 (21:14) update: Modified the contents of Basic and Advanced question.

- **Introduction to Macro & Subroutine:**
  - video: https://youtu.be/XQF7c9myE1Y
  - Hackmd : https://hackmd.io/dlx_c18XSdWER4Y1gbkV2w

- **Lab requirements:**
  - **Basic (70%):**
    - **Description:** Given two points A(x1,y1), B(x2,y2) and x1>x2, y1>y2, design a macro that can calculate the square of the distance between the two points:
      DIST x1, y1, x2, y2, F1, F2
    - **Function:** calculate the square of distance between the two points, the first four arguments map to the value of coordinates x1, y1, x2 and y2. F1 and F2 is the address where the answer stored.
    - **Example test data:**
      After executing DIST 0x05, 0x07, 0x02, 0x03, 0x00 , 0x01:
      Answer = $(0x05 - 0x02)^2 + (0x07 - 0x03)^2$ = **0x0019.**
      Put the high byte of the answer in [0x000], [0x000] = 0x00.
      Put the low byte of the answer in [0x001], [0x001] = 0x19.
    - **Standard of grading:**
      1. We will check whether you use the macros mentioned above. The name and arguments of the macro should be the same as the description.
      2. The actual test data will not be same as example, make sure your code can be execute on any case

  - **Advanced (30%) :**
    - **Description:** Write a subroutine named GP to calculate the sum of the first three terms of a geometric progression. You should use the loop in GP and modify program counter (PCL) instead of using goto and bra. Put the result in address 0x002.

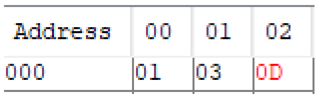
The First term and the common ratio will be positive.

The answer will be limited to 8-bits (0x00~0xFF)

- Example:

    when first term=1 and common ratio=3:

    GP(3) = 1+3+9 = 13



Address	00	01	02
000	01	03	0D

- Standard of grading:
    1. We will check whether you use a subroutine named GP.
    2. You should use rcall.
    3. You should use loop.
    4. The actual test data will not be same as example, make sure your code can be execute on any case
    5. You should modify PCL in the loop instead of using goto and bra.
    6. You cannot use goto and branch instructions which first character is B(ex. BRA,BZ,BN…).
    7. You should put the result in 0x002.

- Bonus (20%):
    - **Description:** Write a subroutine named Hanoitower, use recursion to calculate the minimal number of moves required to solve a Tower of Hanoi puzzle in N disks, and put the result in address 0x000.
    - **Tower of Hanoi:** https://en.wikipedia.org/wiki/Tower_of_Hanoi
    - **Example:**
        1. When N=2, the answer will be 3, put it in [0x000]
        2. When N=3, the answer will be 7, put it in [0x000]
        3. When N=4, the answer will be 15, put it in [0x000]
    - **Standard of grading:**

1. We will check whether you use a subroutine named Hanoitower.
2. You need to use recursion.
3. You should put the result in 0x000.

○ **Hint:**

1. **You don’t need to show the process of moving** ,only have to use recursion to calculate the minimal number of moves.
2. You can use **FSRx** to store the variable used in recursion.