**Q3. Pattern Finding and Cumulative program. (5 points)**

Write a program in 68K assembly code that satisfies the following specifications:

1. Your program should start at the memory location $1000 (hint: use the ORG directive).
2. **Take a one byte hexadecimal value from user input** and save the value at address $A000 (hint: you can consider this input a target value). You should not hardcode the inputs. You may ignore sanity check on corner cases, but assume user inputs are always in an expected format.
3. Read each **byte data** stored in memory between the addresses $6000 and $8000 and compare it to the (byte) data at address $A000 (hint: you can consider the memory between $6000 and $8000 an array).
4. Please define a **LONGWORD**variable called address **Addr1**. In the specified memory range, if the data at address $A000 is found, the **longword address of the data**in memory is saved into variable **Addr1**. If it fails to find the data within the specified memory range ($6000 and $8000), then put Addr1 = $6000, which is similar to the idea of "error code" or "invalid index" in your C++ code.
5. Please define a **WORD** variable called address **Addsum**. Add a series of **bytes** stored in the **512**consecutive memory locations beginning at address Addr1, no matter you find the target data or not. Save the sum value into **Addsum**.
6. There is a chance that the sum might exceed $FFFF, (exceeding the range of word value), so you will also need to store the carry bit if an overflow occurs. Store the **carry bit as a BYTE**variable called **CarryBit**.
7. Print the **Addr1**, **Addsum** and **CarryBit** in the output window.