CS3280
LAB Assignment #2
Due date: Friday, February 23, before 1:00 pm.

You are to design, write, assemble, and simulate an assembly language program which will generate the N_{th} number in the Fibonacci sequence.

What is the Fibonacci Sequence?

You start with initializing the first two numbers of the sequence to 1 ($F_1 = F_2 = 1$). The rule to generate the next number is to add the previous two numbers: $F_N = F_{N-1} + F_{N-2}$.
This results in the following sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ...

Given to you is N as a 1-byte unsigned integer variable with $1 \leq N \leq 255$. Your program has to calculate the N_{th} number in the Fibonacci sequence as a **2-byte number to be stored in RESULT in BIG-ENDIAN format**. For example, if N=10, your answer should be $0037 (the two-byte hex equivalent of 55); if N=24, your answer should be $B520 (the two-byte hex equivalent of 46368).

PLEASE NOTE:
1. Your program should work for any N value, not just the one given.
2. Do NOT use the X or Y registers for storing or manipulating DATA. Only use the X and Y registers for storing/manipulating ADDRESSES.
3. It might be a good idea to use the D-register to add the Fibonacci numbers.
4. You have to use the program skeleton provided for Lab2. Do not change the data section or you will lose points! This means: do not change the 'ORG $B000' and 'ORG $B010' statements or the variable names 'N' and 'RESULT'. You are allowed to change the value assigned to N to simulate different numbers. If you need to define additional variables, please add them after the 'RESULT RMB 2' statement.
5. Your program is NOT allowed to change the number stored in N.
6. Initialize any additional variables that your program needs within the program, NOT with a FCB or FDB in the data section
7. You must terminate your program correctly using the infinite loop structure as shown in class.
8. You do not have to optimize your algorithm or your assembly program for speed.
9. You have to provide a pseudo-code solution. In your pseudo code, do NOT use a for loop, but either a while or a do-until construct to implement a loop. Also, do NOT use any “goto”, “break”, or “exit” statements in your pseudocode.
10. The structure of your assembly program should match the structure of your pseudo code 1-to-1.
11. You do not have to check for overflow when calculating the result.
12. Please pay special attention to the cases of N=1 and N=2.
13. Any assembler or simulator error/warning messages appearing when assembling/simulating your submitted program will result in 50 points lost.

You should test your program with at least two or three different Ns. Figure out the answers (in hex) to be sure that your program is executing correctly.

PLEASE NOTE: Your program will be tested by us using random Ns. If your program does not produce correct result for those random numbers, you will lose up to 50 points.

Your program should include a header containing your name, student number, the date you wrote the program, and the lab assignment number. Furthermore, the header should include the purpose of the program and the pseudocode solution of the problem. At least 85% of the instructions should have meaningful comments included - not just what the instruction does; e.g., don't say "increment the register A" which is obvious from the INCA instruction; say something like "increment the loop counter" or whatever this incrementing does in your program. You can ONLY use branch labels related to structured programming, i.e., labels like IF, IF1, THEN, ELSE, ENDIF, WHILE, ENDWHL, DOUNTL, DONE, etc. DO NOT use labels like LOOP, JOE, etc.

YOU ARE TO DO YOUR OWN WORK IN WRITING THIS PROGRAM!!! While you can discuss the problem in general terms with the instructor and fellow students, when you get to the specifics of designing and writing the code, you are to do it yourself. Re-read the section on academic dishonesty in the class syllabus. If there is any question, consult with the instructor.

Submission:

Electronically submit your .ASM file on Canvas by 1:00pm on the due date. Late submissions or re-submissions (with a 10% grade penalty) are allowed for up to 24 hours (please see the policy on late submission in the course syllabus).

Note:

Because of some inherent lack of reliability designed into computers, and Murphy's law by which this design feature manifests itself in the least convenient moment, you should start your work early. Excuses of the form:

"my memory stick went bad,"
"I could not submit my program,"
"my computer froze up, and I lost all my work;"

should be directed to the memory stick manufacturer, Canvas system administrator, and your local Microsoft vendor respectively.

Grade Requirements and Breakdown

Requirements	Point Value
Program must produce correct answers	50 pts
Program must have good structure and matches pseudo code 1-to-1	30 pts
Good Commenting (including program header and branch labels) and correct Pseudocode	20 pts
Total Points	**100 pts**
Penalties:	
Program does not assemble or is incomplete	-50 pts (No partial credit)
Any assembler or simulator error/warning messages	-50 pts (No partial credit)
No comments at all	-20 pts (No partial credit)
Wrong algorithm implemented	-50 pts (No partial credit)
Late Submission/Resubmission	-10% for up to 24 hours late

Please Note: Submitted programs that won't assemble, produce assembler or simulator warnings/errors, or are incomplete lose 50 points. Be sure to check/assemble/simulate your code one last time before you submit your assignment!!!!