CS/ECE 3280
LAB Assignment #3
Due date: Friday, October 7, before 2:00 pm.

You are to design, write, assemble, and simulate an assembly language program which will generate the sum of integer squared from 1 to N (aka *Square Pyramid Number*).

The sum of integer squares is formed by calculating the following sum:

RESULT = $1^2 + 2^2 + 3^2 + 4^2 + \ldots + N^2$, with $1 \leq N \leq 254$

Given to you is N as a 1-byte **unsigned** integer variable with **$1 \leq N \leq 254$**. Your program has to calculate the sum of squares as a **4-byte number** to be stored in variable RESULT in BIG-ENDIAN format. For example, if N=254, your answer should be $0053D77F. To verify your program, you can use the following formula to calculate the sum of integer squares:

$1^2 + 2^2 + 3^2 + 4^2 + \ldots + N^2 = N\,(N+1)\,(2N+1) / 6$

PLEASE NOTE:

1. N is a 1-byte **unsigned** integer variable with **$1 \leq N \leq 254$**. Thus, you don't have to check if N = 0, or N =255.
2. Your program should work for any N value, not just the one given.
3. You have to use the algorithm given in Lab2 to calculate individual squares, and the summation given above to calculate the sum of squares. This will result in an implementation using two nested loops (the inner loop calculates each individual square, while the outer loop calculates the sum of these squares); using any other algorithm, implementation, multiplication, the above formula, or the MUL instruction instead is NOT allowed.
4. Your program is NOT allowed to change the number stored in N.
5. You have to use the program skeleton provided for Lab3. Do not change the data section or you will lose points! This means: do not change the 'ORG $B000' and 'ORG $B010' statements, 'N FCB'. 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 4' statement.
6. You are allowed to use parts of your LAB2 or parts of the official LAB2 solution.
7. You must terminate your program correctly using the STOP instruction as shown in class.
8. The program must only have one exit point (i.e., only one STOP instruction at the end of the program is allowed).
9. You do not have to optimize your algorithm or your assembly program for speed.
10. 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 structure to implement a loop. Also, do NOT use any "goto", "break", "exit", or "return" statements in your pseudocode.
11. The structure of your assembly program should match the structure of your pseudo code 1-to-1 (e.g., if the pseudo code shows a while structure, your assembly program should also have a while structure).
12. Make sure that you implement any if-then, if-then-else, while, or do-until structures the way you learned it in class. Incorrectly implemented structures will result in points lost.
13. Any assembler or simulator error/warning messages appearing when assembling/simulating your submitted program will result in up to 50 points lost.

You should test your program with at least five different N values; for example:

A. N = 1 -> RESULT = $00000001
B. N = 5 -> RESULT = $00000037
C. N = 100 -> RESULT = $000529AE
D. N = 200 -> RESULT = $0028FEEC
E. N = 254 -> RESULT = $0053D77F

PLEASE NOTE: Your program will be tested by us using random number pairs. If your program does not produce correct results for those random pairs, you will lose up to 25 points (see grading guidelines below).

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. **Remember:** labels need to be unique. So, for example, if you have two if-then structures, you should use the labels IF,THEN, ENDIF for the first structure and IF1,THEN1, ENDIF1 for the second one.

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 2 :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

The assignment is worth 100 points. The following deductions will be made (maximum deduction of 100 pts):

Correct Pseudocode

- Pseudocode incomplete or missing: -50 points
- wrong algorithm implemented (two nested loops needed): -50 points
- re-submitting an older lab: -100 points
- submitted program not related to lab assignment: -100 points
- for-loop used in pseudocode: -15 points
- "break/exit/return/goto" used in pseudo-code: -10 points

Program must produce correct results

- program does not produce correct result for certain N values: up to -25 points

Program must have good structure

- program does not assemble or is incomplete: -50 points
- assembler or simulator error/warning messages during assembly/simulation: -25 points
- multiple STOP instructions used: -10 points
- program changes N variable: -5 points
- structure X incorrectly implemented: -5 points for each structure
- hardcoded addresses used in program: -5 points

Program must match pseudo code 1-to-1

- program structure does not match pseudo-code (program implements structure X, but different or no structure shown in pseudo code; or vice versa): -5 points for each structure
- branch for signed numbers used but variables unsigned; or vice versa: -5 points for each branch
- conditional branch used does not match pseudo-code condition: -5 points for each branch

Good Commenting

- no program comments at all: -20 points
- program not commented enough: -10 points
- program description missing: -5 points
- incomplete program header: -5 points
- incorrect branch labels used: -5 points

Note: This list is by no means comprehensive but only lists the most common deductions.