CS3280
LAB Assignment #3
Due date: Friday, October 4, before 12:00 pm.

You are to design, write, assemble, and simulate an assembly language program which will calculate **Euclid's Greatest Common Divisor** of two **unsigned** 1-byte numbers NUM1 and NUM2.

Euclid's Algorithm finds the greatest common divisor (GCD) of two positive integers. Given integers A and B, calculate R_1 = A MOD B. If $R_1 = 0$, then B is the GCD of A and B; otherwise, calculate R_2 = B MOD R_1. If $R_2 = 0$, then R_1 is the GCD of A and B. Otherwise repeat by calculating $R_{n+1} = R_{n-1}$ MOD R_n until $R_{n+1} = 0$, which makes R_n the GCD of A and B.

Note that the divisor and remainder at level n become the dividend and divisor at level n+1.

Example: Find the GCD of 64 and 36.

```
64 / 36 = 1, remainder = 28
36 / 28 = 1, remainder = 8
28 / 8 = 3,  remainder = 4
8 / 4 = 2,   remainder = 0
```

The algorithm halts when the remainder is zero, showing the last divisor to be the GCD. So the GCD of 64 and 36 is 4. If the two integers have GCD = 1, we say they are relatively prime, meaning they have no common (non-trivial) divisors.

PLEASE NOTE:

1. NUM1, NUM2, and GCD are 1-byte **unsigned** integer variables with **NUM1 >0** and **NUM2 > 0**. Thus, you don't have to check if NUM1 or NUM2 is 0.
2. Your program should work for any NUM1 and NUM2 values, not just the ones given.
3. Using the DIVL instruction to calculate the GCD is NOT allowed. **You need to use the above-mentioned algorithm, resulting in two nested loops**. Any implementation that does not use two nested loops will result in lost points.
4. Your program is NOT allowed to change the numbers stored in NUM1 and NUM2.
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 or the variable names 'NUM1', 'NUM2' and 'GCD'. You are allowed to change the value assigned to NUM1 and NUM2 to simulate different numbers. If you need to define additional variables, please add them after the 'GCD RMB 1' 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 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 structure to implement a loop. Also, do NOT use any "goto", "break", "exit", or "return" statements in your pseudocode.
10. 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).
11. 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.

12. Any assembler or simulator error/warning messages appearing when assembling/simulating your submitted program will result in 25 points lost.

You should test your program with at least five sets of decimal data:

A. NUM1= 182, NUM2= 248 -> GCD = 2
B. NUM1= 248, NUM2= 186 -> GCD = 62
C. NUM1= 125, NUM2= 250 -> GCD = 125
D. NUM1= 1, NUM2= 100 -> GCD = 1
E. NUM1= 255, NUM2= 251 -> GCD = 1

PLEASE NOTE: Your program will be tested by us using random NUM1/NUM2 pairs. If your program does not produce a correct result for those 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 12:00pm 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: -50 points
- for-loop used in pseudocode: -10 points
- "break/exit/return/goto" used in pseudo-code: -10 points

Program must produce correct results

- program does not produce correct result for certain NUM1/NUM2 pairs: 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
- DIVL instruction used in program: -50 points
- program changes NUM1/NUM2 variables: -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 vise versa: -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.