

# Program #1

---

**Due** Jan 16 by 11:59pm    **Points** 50    **Submitting** a file upload    **File Types** asm

---



Due: Week 2, Sunday 11:59 PM Pacific USA time zone

## Objectives

1. Introduction to MASM assembly language
2. Defining variables (integer and string)
3. Using library procedures for I/O
4. Integer arithmetic

## Description

Write a MASM program to perform the tasks listed below. Test your program to ensure that it functions correctly.

1. Display your name and program title on the output screen.
2. Display instructions for the user.
-  3. Prompt the user to enter two numbers.
-  4. Calculate the sum, difference, product, (integer) quotient and remainder of the numbers.
5. Display a terminating message.

## Requirements

1. The **main** procedure must be divided into sections:
  - introduction
  - get the data
  - calculate the required values
  - display the results
  - say goodbye
2. The results of calculations must be stored in named variables before being displayed.

3. The program must be fully documented. This includes a complete header block for identification, description, etc., and a comment outline to explain each section of code.
4. Submit your text code file (.asm) to Canvas by the due date.

## Notes

1. A program shell (template) is available on the course website. See the Week 1 Basics page in Modules.
2. You are not required to handle negative input or negative results.
3. You have a limited number of late days. Try not to use these on the first program.
4. To create, assemble, run, debug, and modify your program, follow the setup instructions available within Canvas on the Syllabus → Tools page.
5. Find the assembly language instruction syntax in the textbook.
6. Documentation for the Irvine library procedures is provided in the textbook.

### Example Program Operation

```
Elementary Arithmetic by Wile E. Coyote
Enter 2 numbers, and I'll show you the sum, difference, product, quotient, and remainder.
First number: 37
Second number: 5
37 + 5 = 42
37 - 5 = 32
37 x 5 = 185
37 / 5 = 7 remainder 2
Press any key to continue. Pressed? Bye!
```



## Extra Credit Options (original definition must be fulfilled)

1. (1 pt) Validate the second number to be less than the first.
2. (1 pt) Display the square of each number. Recall that the square of a number is obtained by multiplying a number by itself. For example:

```
Square of 37 = 1369
Square of 5 = 25
```

To ensure you receive credit for any extra credit options you did, you must add one print statement to your program output PER EXTRA CREDIT which describes the extra credit you chose to work on. You will not receive extra credit points unless you do this. The statement must be formatted as follows...

```
--Program Intro--  
**EC: DESCRIPTION  
--Program prompts, etc--
```

For example, for extra credit option #1:


```
Elementary Arithmetic by Wile E. Coyote  
**EC: Program verifies second number less than first.  
Enter 2 numbers, and I'll show you the sum, difference, product, quotient, and remainder.  
First number: 7  
Second number: 9  
The second number must be less than the first!  
Impressed? Bye!
```

## Program 1 Rubric



Criteria	Ratings			Pts
Preliminaries - Files Correctly Submitted Submitted file is correct assignment and is an individual .asm file.	<b>1 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>		1 pts
Preliminaries - Program assembles, links Submitted program assembles and links without need for clarifying work for TA and/or messages to the student. This assumes the program is actually an attempt at the assignment. Non-attempts which compile/link earn no points.	<b>2 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>		2 pts
Documentation - Identification Block - Header Name, OSU Email, Course number, Program number, Date, etc as per syllabus are included in Identification Block <div>▶</div>	<b>2 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>		2 pts
Documentation - Identification Block - Program Description Description of functionality and purpose of program is included in identification block.	<b>2 pts</b> <b>Full Marks</b>	<b>1 pts</b> <b>Lacking detail</b> Description is present but is lacking in detail with regard to functionality.	<b>0 pts</b> <b>No Marks</b>	2 pts

Criteria	Ratings			Pts
<p>Verification - Program Executes</p> <p>Program executes and makes some attempt at the assigned functionality.</p>	<p><b>5 pts</b> <b>Full Marks</b></p>	<p><b>2 pts</b> <b>Failed attempt</b></p> <p>Program is an attempt at the correct assignment but simply does not run.</p>	<p><b>0 pts</b> <b>Wrong Program</b></p> <p>Program executes but is either the incorrect program or some quickly mashed together nonsense, submitted only to 'get a few points'</p>	<p>5 pts</p>
<p>Completeness - Displays programmer name</p> <p>Program prints out the programmer's name.</p>	<p><b>1 pts</b> <b>Full Marks</b></p>		<p><b>0 pts</b> <b>No Marks</b></p>	<p>1 pts</p>
<p>Completeness - Displays Introduction</p> <p>Program displays the program introduction.</p>	<p><b>2 pts</b> <b>Full Marks</b></p>		<p><b>0 pts</b> <b>No Marks</b></p>	<p>2 pts</p>
<p>Completeness - Prompts user to input data</p> <p>Program outputs a data request to user e.g. "Enter a number" twice.</p>	<p><b>1 pts</b> <b>Full Marks</b></p>		<p><b>0 pts</b> <b>No Marks</b></p>	<p>1 pts</p>
<p>Completeness - Gets data from user</p> <p>Utilizes ReadDec or ReadInt, gets user-input data, and saves data to some memory variable (non-register).</p>	<p><b>2 pts</b> <b>Full Marks</b></p>		<p><b>0 pts</b> <b>No Marks</b></p>	<p>2 pts</p>

Criteria	Ratings		Pts
Completeness - Displays Results Program displays results in the form of (X + Y = M   X - Y = N   X * Y = O   X / Y = Q remainder R)	2 pts Full Marks	0 pts No Marks	2 pts
Correctness - Calculations are correct Calculations are all correct. Lose 1 point per incorrect calculation (Sum   Difference   Product   Quotient   Remainder)	5 pts Full Marks	0 pts No Marks	5 pts
Correctness - Original User Data Unchanged Original variables holding user-entered data (num1, num2) remain unchanged by all calculations and still hold original user-entered  at end of execution.	2 pts Full Marks	0 pts No Marks	2 pts

Criteria	Ratings				Pts
<p>Requirements - Modularized Code Blocks</p> <p>Main procedure is separated into functional sections, each of which is described by comments.</p>	<p><b>5 pts</b> <b>Blocks with Headers</b></p>	<p><b>3 pts</b> <b>Blocks without Headers</b></p> <p>Program is separated into logical blocks but those blocks are poorly commented</p>	<p><b>3 pts</b> <b>Headers without Blocks</b></p> <p>Program is not visibly separated into logical blocks with whitespace, but limited headers do indicate some organizational effort.</p>	<p><b>0 pts</b> <b>No Marks</b></p>	<p>5 pts</p>
<p>Requirements - Results stored in named variables</p> <p>Results of calculations are stored in memory in discrete variables.</p> <p>Lose 1 point per missing variable usage (Sum   Difference   Product   Quotient   Remainder)</p>	<p><b>5 pts</b> <b>Full Marks</b></p>		<p><b>0 pts</b> <b>No Marks</b></p>		<p>5 pts</p>
<div data-bbox="79 976 149 1036">▶</div> <p>ing Style - In-line Comments</p> <p>In-line comments and block headers describe functionality of program flow. Should mirror the style guide image.</p>	<p><b>5 pts</b> <b>Full Marks</b></p>		<p><b>0 pts</b> <b>No Marks</b></p>		<p>5 pts</p>

Criteria	Ratings			Pts
Coding Style - Appropriately Named Identifiers  Identifiers named so that a person reading the code can intuit the purpose of a variable, constant, or label just by reading its name.	3 pts Full Marks	1 pts Partial  Some identifiers are named well, with others having no relevance to their functionality.	0 pts No Marks	3 pts
Coding Style - Readability  Program uses readable white-space, indentation, and spacing as per the Indentation Style Guide. Logical sections are separated by white space.	5 pts Full Marks	3 pts Marginally Readable  Program is marginally readable but lacks proper alignment and white space.	0 pts No Marks	5 pts
Late Penalty  Remove points here for late assignments. (Enter negative point value)	0 pts Full Marks		0 pts No Marks	0 pts
<div>▶</div> No prohibited macros (see lecture 6) REJECT ASSIGNMENT  None of the assignments are allowed to use .IF, .IFELSE, or related macros. Only actual assembly code.	0 pts Full Marks		0 pts No Marks	0 pts



Criteria	Ratings		Pts
<p>Extra Credit (1pt) - validate that num2 is less than num1</p> <p>Assuming that num1 and num2 are non-zero positive integers, the code must generate an error if num2 is greater than or equal to num1.</p>	<p><b>0 pts</b> <b>Full Marks</b></p>	<p><b>0 pts</b> <b>No Marks</b></p>	0 pts
<p>Extra Credit (1pt) - display mathematical square of the user's numbers</p> <p>The code displays the square value of the numbers that the user entered (see assignment for explanation).</p>	<p><b>0 pts</b> <b>Full Marks</b></p>	<p><b>0 pts</b> <b>No Marks</b></p>	0 pts
Total Points: 50			

