Individual Assignment 1: Implementation of Cross-Correlation on 1-D Arrays Using Intel Assembly

Important notes:

- In this homework you will work with NASM on an 32-bit Linux, same as what you have used in class and recitations. Assignments, which use another assembler or incompatible with 32-bit Linux platform will not be evaluated.
- This is an individual homework, not a group project.

Description of Assignment

In this assignment, you will implement a function that performs full cross-correlation on two 1-D arrays.

A function written in Intel assembly language will be called from a program in C as shown below. The function you implemented in assembly must be fully compatible with given prototype.

```
int cross_correlation_asm_full(int* arr_1, int size_1, int* arr_2, int size_2,
int* output, int output_size);
```

Parameters:

- *int* arr_1*: First input array.
- *int size_1*: Number of elements in the first array.
- *int* arr_2*: Second input array.
- *int size_2*: Number of elements in the second arrays.
- *int* output*: Output array, you will fill this array with calculated values.
- *int output_size*: Number of elements the output array is going to have.

Return value:

- Number of non-zero elements in the output array.

About your implementation:

- You are not required to check edge cases such as where one of the arrays or both are empty. Just assume the given inputs are always valid.
- Input arrays will be consist of natural numbers only (x >= 0 and x is integer)
- Do not perform any normalization after the cross-correlation operation. Sample inputs and outputs will be given in next sections.
- You are not required to check for any overflow, numbers given to you will be small enough to perform cross-correlation operation on 32-bit signed integers.
- You are not allowed to modify given C code.

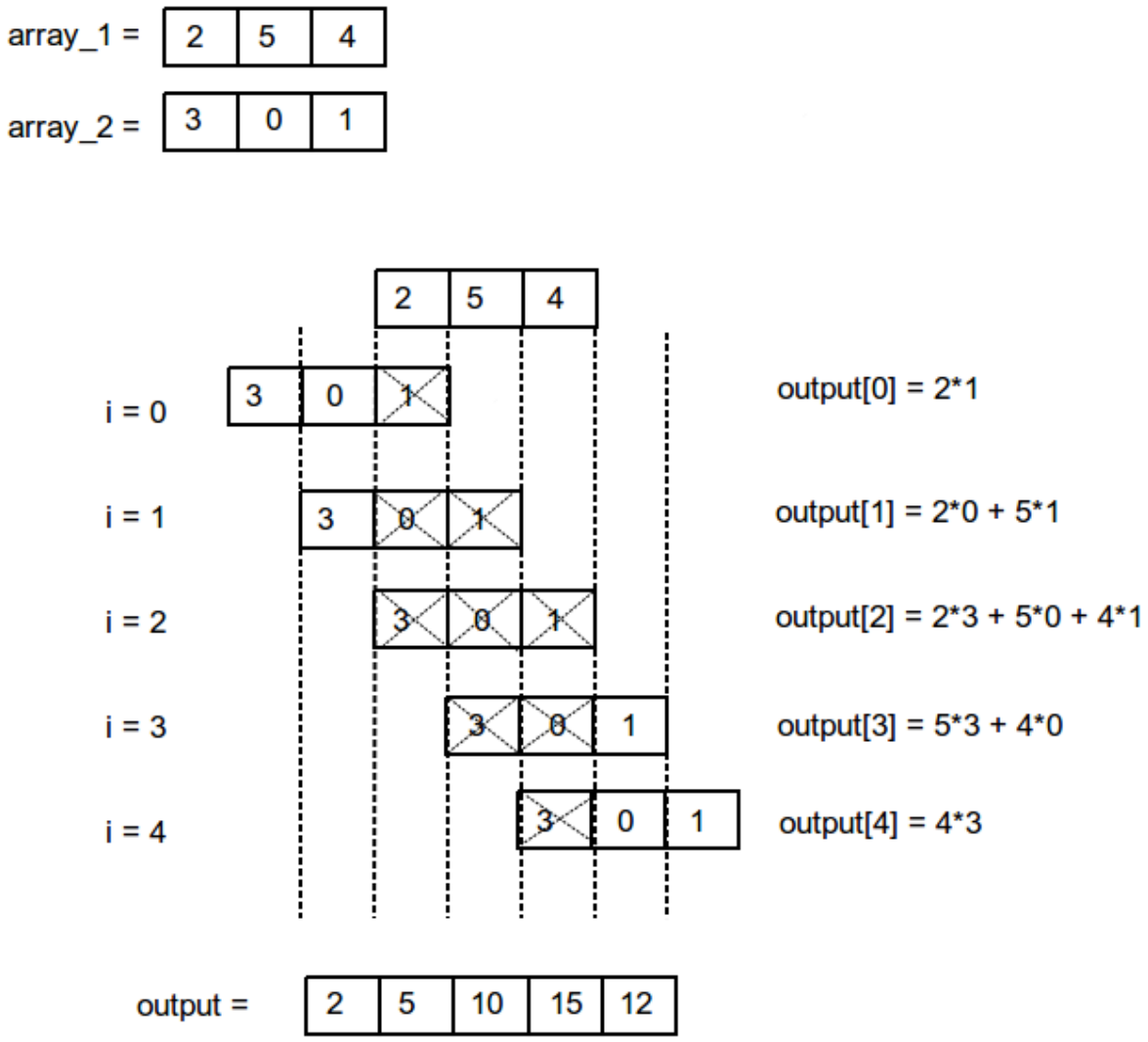
You will be given a C code implementing main function, which reads the input arrays from a text file before calling `cross_correlation_asm_full` function using these arrays as inputs. After the cross correlation operation, it writes the results to an output file. These input arrays will be read from file "arrays.txt". The first line of the file contains number of arrays in the file, in case you want to modify it and test with other arrays. The file may contain more than 2 arrays. In that case, the given C program will call `cross_correlation_asm_full` function for each possible combination.

Additional information explaining the cross correlation operation is given below:

Cross-Correlation Operation

Cross-correlation is one of the ways to measure similarity of two functions/signals in signal processing. Although, it can operate on any two continuous/discrete arrays containing real or complex values, in this assignment, you are just going to implement this function for discrete arrays that consist of natural numbers only.

It is also known as sliding dot product. It can be performed by fixing one of the arrays, sliding the other one step by step and multiplying overlapping numbers. A numeric example is given below explaining the cross-correlation step-by-step:



This output is the result of "**full**" cross-correlation. There are other types such as "**valid**" and "**same-padding**" but those are not related to assignment.

Sample Cross-Correlation Inputs/Outputs

array_1 = {2, 5, 4}
array_2 = {3, 0, 1}
output = {2, 5, 10, 15, 12}

array_1 = {2, 5, 4}
array_2 = {2, 5, 4, 1}
output = {2, 13, 34, 45, 30, 8}

array_1 = {2, 5, 4}
array_2 = {3, 0, 1, 2}
output = {4, 12, 13, 10, 15, 12}

array_1 = {2, 5, 4}
array_2 = {7}
output = {14, 35, 28}

array_1 = {2, 5, 4}
array_2 = {6}
output = {12, 30, 24}

array_1 = {2, 5, 4}
array_2 = {19, 11, 0}
output = {0, 22, 93, 139, 76}

array_1 = {2, 5, 4}
array_2 = {0, 0, 0}
output = {0, 0, 0, 0, 0}

array_1 = {3, 0, 1}
array_2 = {6}
output = {18, 0, 6}

array_1 = {7}
array_2 = {6}
output = {42}

array_1 = {7}
array_2 = {19, 11, 0}
output = {0, 77, 133}

array_1 = {2, 5, 4, 1}
array_2 = {3, 0, 1, 2}
output = {4, 12, 13, 12, 16, 12, 3}

Submission Details

You are required to implement the given function in Intel assembly. The function implementation must fully conform to the provided prototype since it is expected to be linked to the main program implemented in C which will be provided through the Ninova system. The main program will handle all input/output operations, so you do not need to read or write anything from the assembly. Your submissions will be tested automatically via a script, therefore, you should use the provided C program without modifying it. Your program is going to be tested with given sample inputs and additional ones.

You are required to submit your assembly language source code file(s) through the Ninova system as a zip file.

Any form of cheating or plagiarism will not be tolerated. This includes actions such as, but not limited to, submitting the work of others as one's own (even if in part and even with modifications) and copy/pasting from other resources (even when attributed). Serious offenses will be reported to the administration for disciplinary measures.