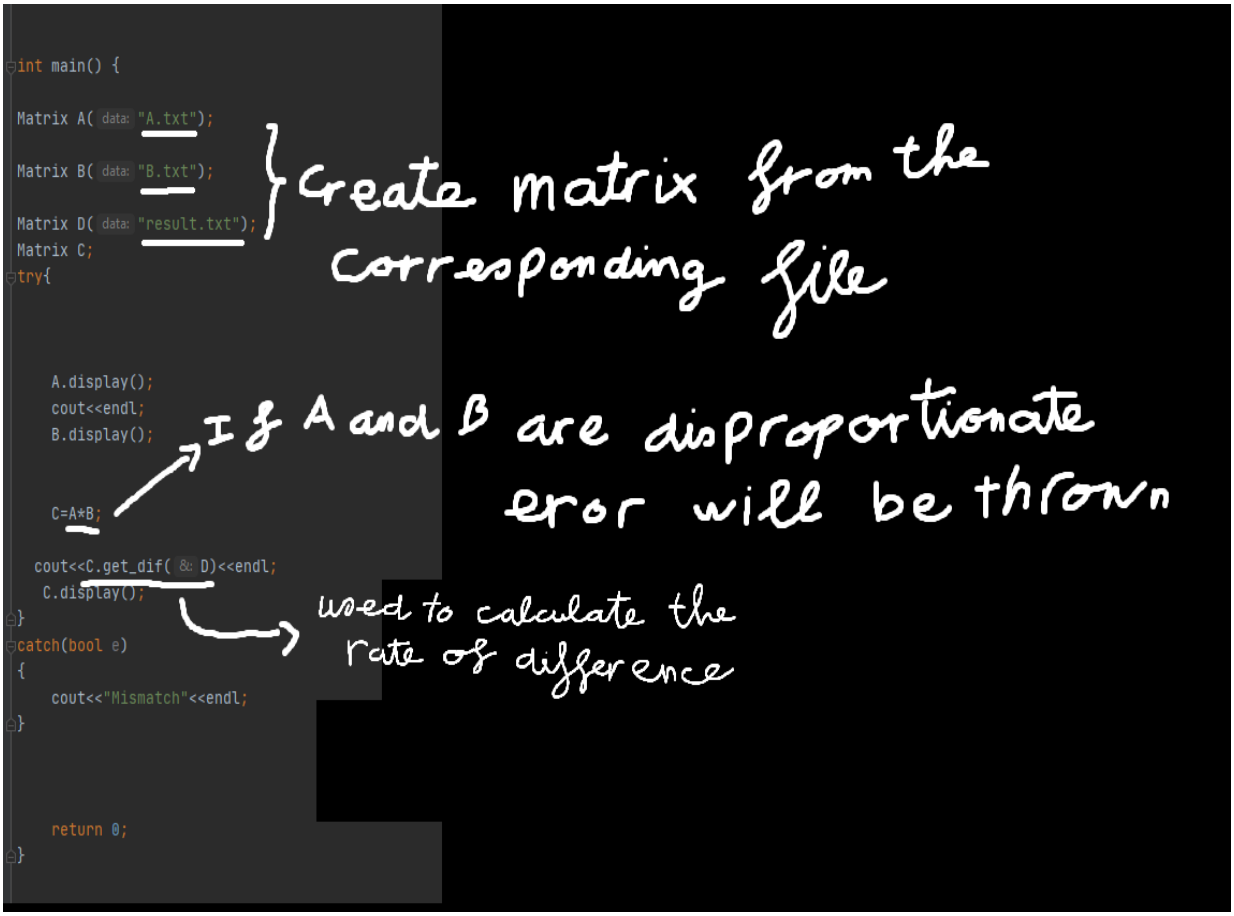
Assignment CA report

Ex1:

Explanation of the use of the program:

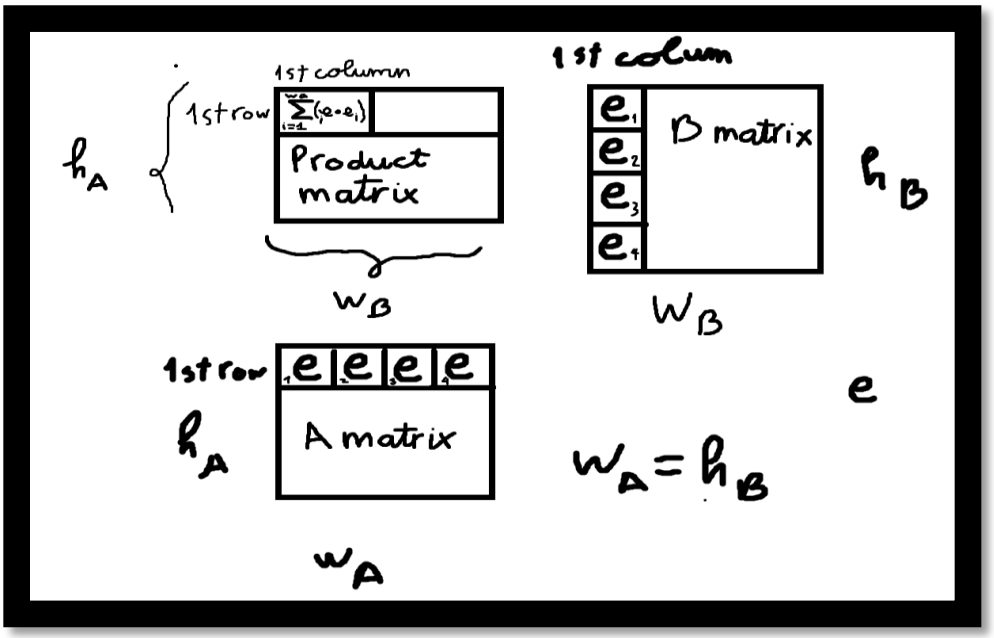
For this part of the assignment the program is writent in c++. Its inputs are three files (A.txt, B.txt, result.txt), each contains one matrix with the matrix dimensions specified before the matrix. The matrix stored in A.txt and B.txt are used for constructing the multiplier matrix (A matrix) and multiplicand matrix (B matrix), if the dimensions of A matrix and that of B matrix fail to meet the rule of matrix multiplication an error will be thrown, otherwise the program will perform matrix multiplication for these two matrices and the product will be then compared with the D matrix, which was initially created from result.txt. Using get_diff() method to calculate the rate of difference between the D matrix and the product matrix of A and B.



Algorithm for matrix multiplication:

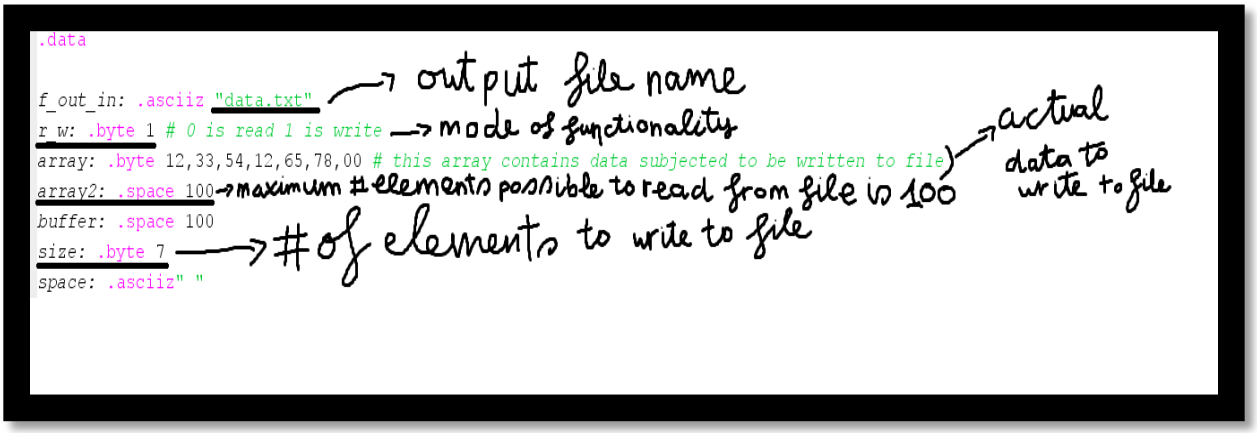
To perform matrix multiplication, first of all, the multiplier matrix and multiplicand matrix are represented by two 2D arrays with the height and width of each array correspond to the height and width of each matrix, now generate the multiplication mechanism with three loops nested in each other, the outer most loop is responsible for keeping track of the rows of A matrix, the second loop is accounted for the order of the columns of the B matrix, the inner most loop is used to preform element wise multiplication for the elements in A matrix particular row with B matrix particular columns and then sum up these products and assign the value to an element of product matrix which has its address to be (x,y) where x= the position of the current selected row of A matrix and y

is the position of the current selected column of B matrix. The process keeps repeating until all elements of product matrix have been filled with new value.



Ex2:

This program aims at assisting users to read data from file and store them to a certain unit in the program or perform the reverse operation. Users can specify the destination file to write to as well as the size and the data subjected to be transferred, however on the read from file mode there is a limit of number of characters possible to obtain from file, maximum 100 characters to be specific (this is due to the design of the procedure). The following figure will illustrate what parameters users can modify to command the program to function according to their wish:



To select the file participating during program's operation, user can modify the file name specified right after the parameter: f_out_in .

To choose the operation mode for the program, user must adjust the value of r_w with 0 to be read from file mode and 1 to be write to file mode.

The parameter array stores actual integers that are subjected to be written to file, user can type integers that they wish to write to file into this array (notice the value of the eligible integers is in the range from 0 to 127).

The last parameter user can modify is size this parameter will decide the actual number of integers in array that will be written to file starting from left to right.

Algorithm of write to file mode :

To write to file, the address of array will first be loaded into a register, which will then be passed to and processed by a function to extract each element of the array, after extracting a particular element from the array, the element must be converted to asccii format by extracting each digit of the number (using % 10) and add 48 to each digit to convert to

asccii value, after that, store each digit to the stack and the process repeats until all the digits have been extracted, once the stack has fully contained the ascii encoded version of a particular element, the program will execute a syscall instruction specilized for file IO operation and write to file the contents initially stored in the stack obviously the number of elements written to file is the number of digits. The number of elements that are proceesed is equal to the value the size parameter.

Algorithm of read from file mode :

To read from file we first access the file using syscall instruction and obtain the first 100 characters available in the file then store them in array2, array2 will store values in the format such as: 50 49 51 32 50 53 54 32 ….

With the above series of integers, it can be decoded to “213 space 256 space” each element represents a digit in ascii code with 32 is “space” in ascii code, to convert them back to integers each element from array2 is extracted and reduced its value by 48, as long as the next element is not 32 the number is multiplied by 10 and get added to a register, the register accumulates the array elements until the next element has value 32 the register will transfer its value to the corresponding position in buffer and move to the next position. The process repeats until the last element of array2.

Ex3:

This exercise focuses on creating 2D storages for data using dynamic allocation method.

Users can interact with the program by modifying the following parameters:

```
.data
#specifing the size of the array its individual elements
array_size: .byte 5
elem_size: .byte 12,2,3,43,10
space: .asciiz" "
endl: .asciiz"\n"
.text
```

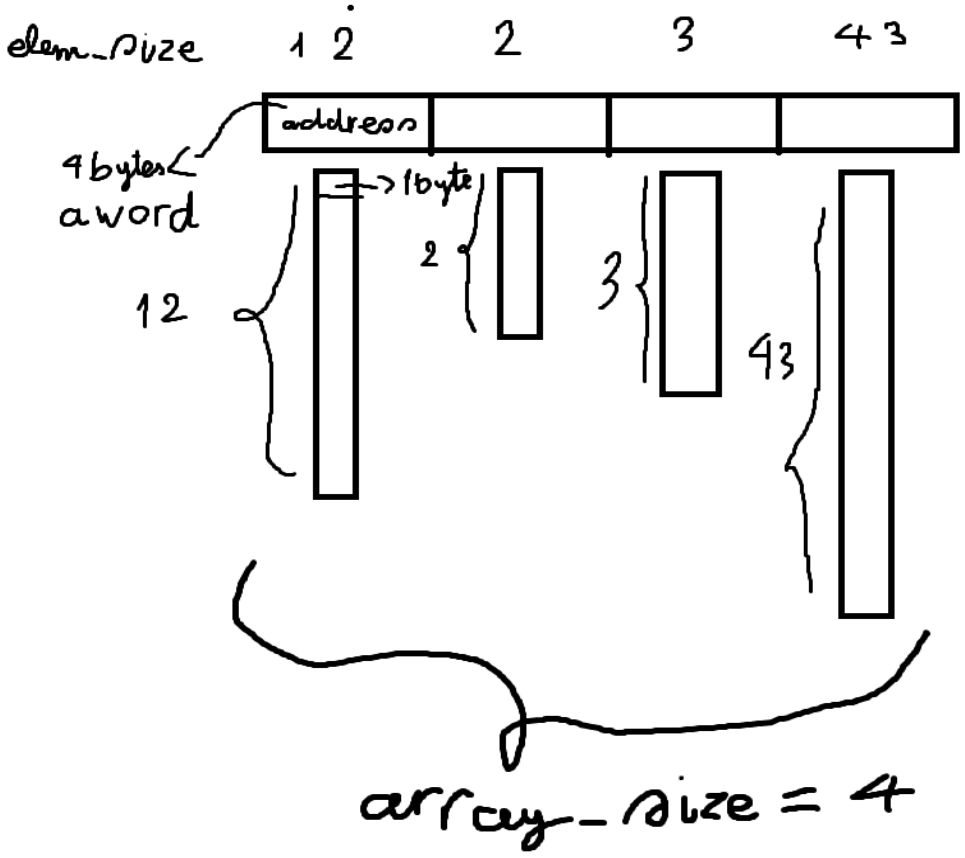
With array_size parameter, users are able to modify the size of the the main array by changing the number of bytes to allocate.

The main array is used to store the addresses of the sub array and the sizes of the subarray are specified by the following parameter, elem_size: this parameter holds the address of an array with its elements are the sizes of sub arrays, by changing theses values users can adjust the sizes of the sub arrays according to their wish. However, there is an exception, of the total number of elements exceeds 65536 an error will be thrown, nevertheless there is still space being allocated in this situation.

Algorithm to construct 2D storage structure:

First of all, an array with each element has the size of a word will be dynamically allocated and the number of elements equals the value of array_size.

Now we need a loop to go through each element of the array and at each element, dynamically allocates a sub array whose storage unit is in byte and the sub array size is the value of the corresponding element in array_size. The following figure helps visulize the algorithm idea:



Ex4:

This exercise uses assembly language to implement matrix multiplication.

User interaction:

Users can interact with the program by adjusting the following parameters:

```
.data

matrixA: .word 10, 5, 0
matrixB: .word 5, 5, 1
matrixC: .space 12
space: .asciiz" "
endl: .asciiz"\n"
error_message: .asciiz"dimension mismatch!\n"
.text
```

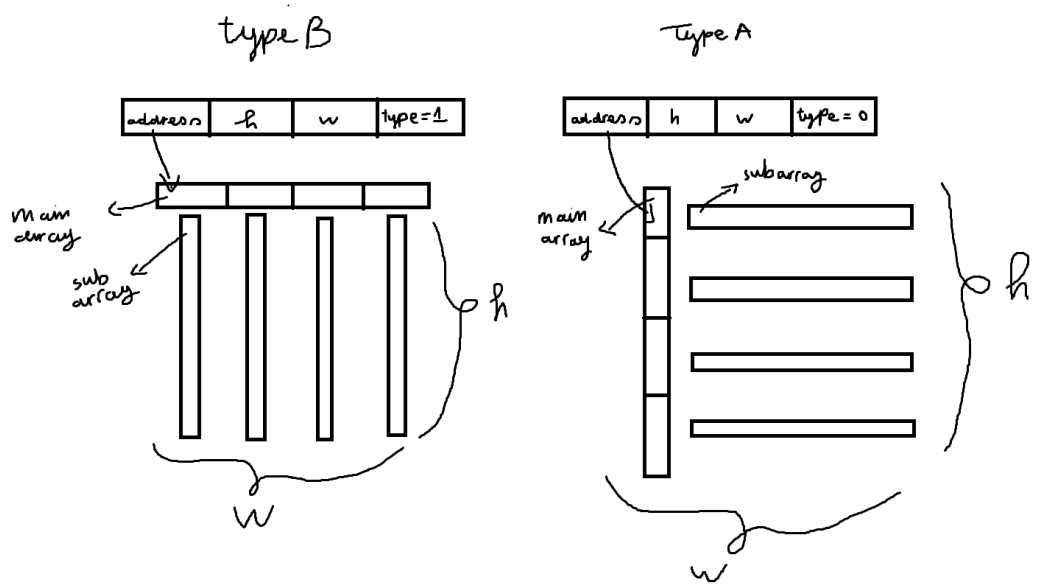
For the convenience of constructing matrix multiplication algorithm, two types of matrices are defined, A and B, users can choose their own dimension for the matrices by modifying numbers in matrixA and matrixB , the first parameter in matrixB and matrixA is the height of the matrix and the second parameter represents the width of the matrix, the third parameter is the type of the matrix (by default the value for matrix A is 0 and matrix B is 1, users should not change this value in order for the program to function properly).

Procedure explanation:

when matrixA and matrixB are created the program will perform matrix multiplication if the width of matrix A is not equal to the height of the matrix B, the program will return with error code –1 which is assigned to $v0 register.

Algorithm explanation:

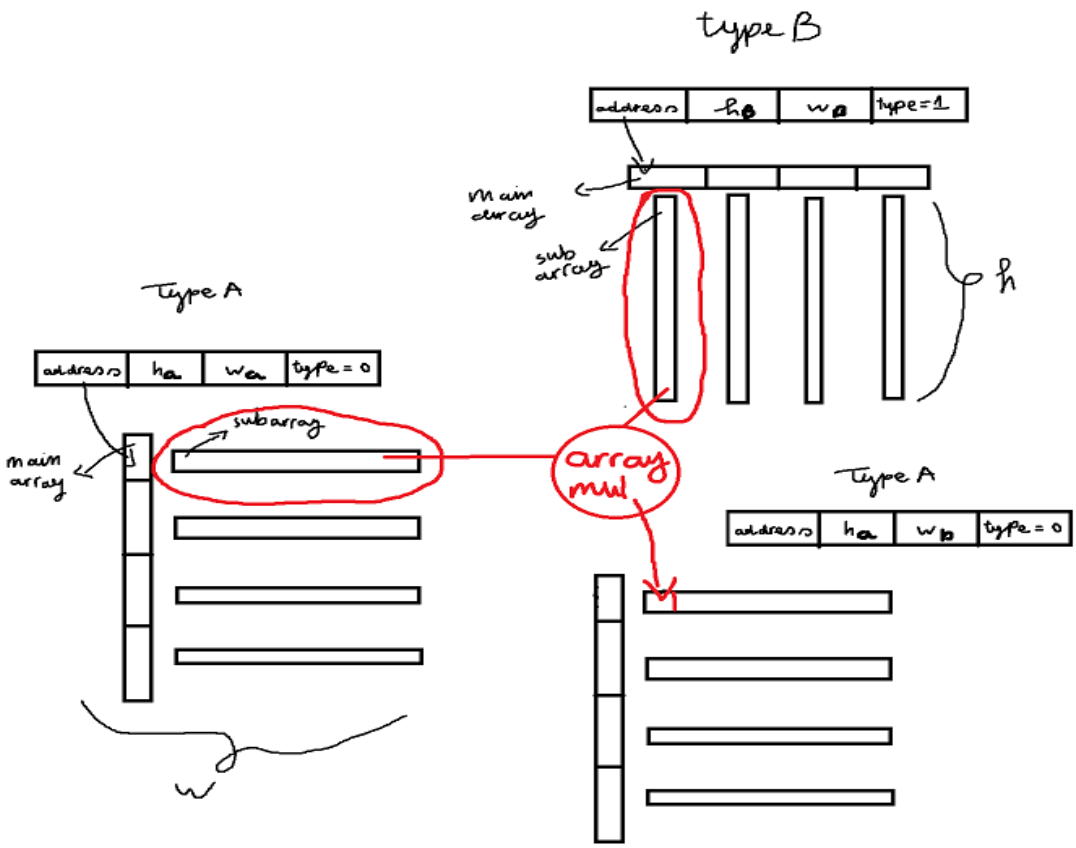
Before we come to clarifying the algorithm for matrix multiplication, we first look at the definition of type A and type B matrices, the following figure will help visualize the concept:



The difference between type A and type B is that with type A matrix, the height will correspond to the length of the of the main array, while its width is the length of the sub arrays, notice that all sub arrays have the same length. In type B matrix the opposite regulation is applied.

To implement matrix multiplication procedure, we first create a loop to trace along the main array of matrix A, a register, let's call Row, is responsible to keep the address stored in the current index of the main array. Another loop nested inside the first loop will trace along the main array of matrix B and another register, let's call Column, will store the value of the current index of the main array and at each iteration Row and Column will go to through a procedure called array_mull to perform

element wise multiplication, this procedure will calculate the product of each pair of elements with each element is from a different array from the other and they are paired up in the cardinal order for instance 1st element of Row will go with the 1st element of Column and so on. The product of each pair will then be summed up and assigned to an element in the product matrix illustrated in the following figure:



The process repeat until all element in the product marix have been filled with new values.

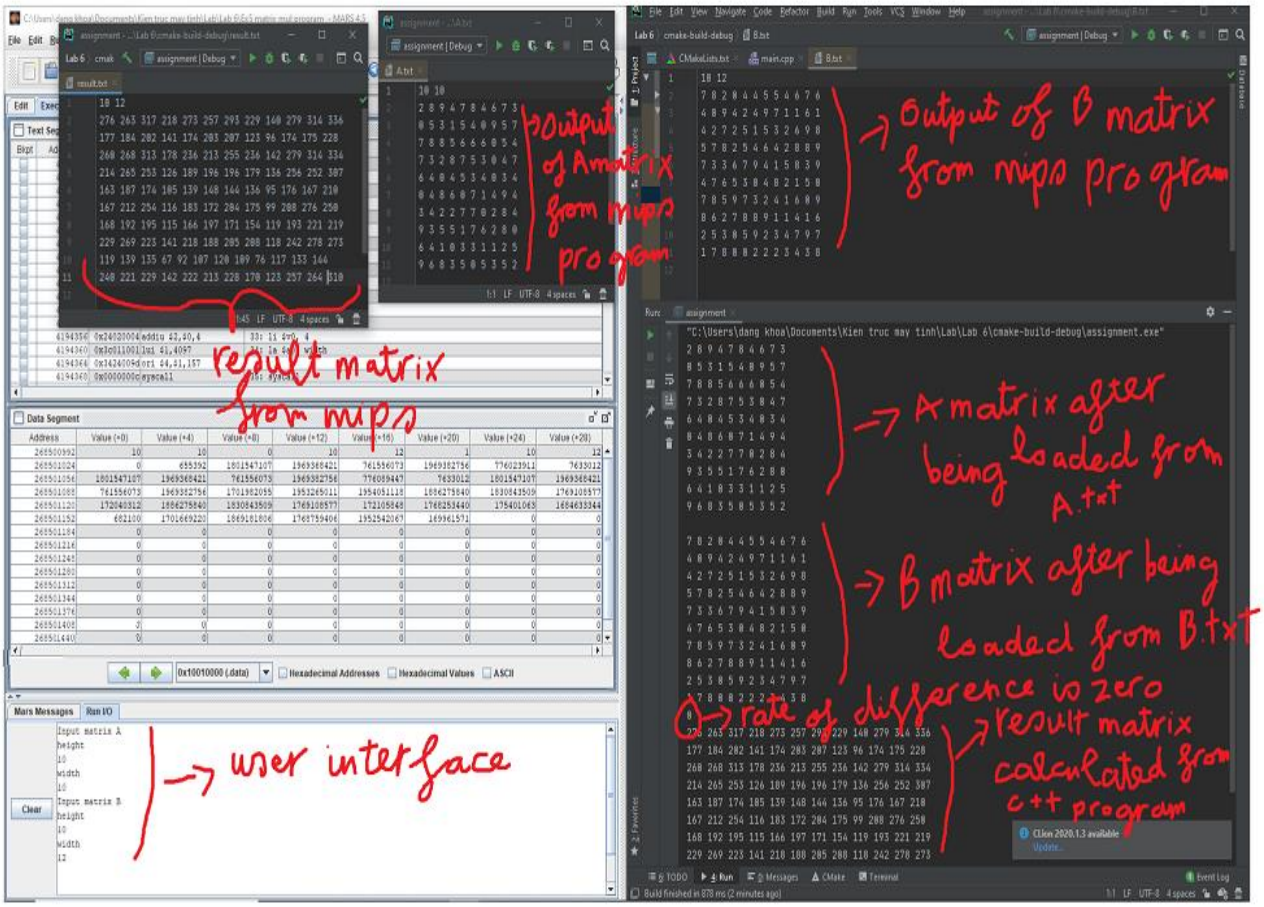
Ex 5:

This exercise employs the procedures of all the above exercises to create a program where users input the dimensions of 2 matrices, with the previously given data, the program will create 2 matrices with the value of each element to be randomly generated in the range from 0 to 9. This is possible due to the random generator procedure implemented in the code. Once the two matrices have been constructed each will be loaded into a separated files (A.txt for matrix A and B.txt for matrix B) with format stated in Ex1.

Multiplication procedure:

If their dimensions are incompatible for matrix multiplication, the program will terminate with an error massage stated “dimension mismatched”, otherwise another matrix will be created with its height is of matrix A and the width is of matrix B. After this process the program will perform matrix multiplication with its functionality to be similar to that in Ex4. Once the procedure is complete the new matrix will be transferred to a third file namely result.txt. The c++ program introduced in Ex1 will be used to check if the result is correct as stated in Ex1.

The following pictures are demonstrations of the functionality of the program:


result matrix from mips
output of A matrix from mips program
Output of B matrix from mips program
A matrix after being loaded from A.txt
B matrix after being loaded from B.txt
rate of difference is zero
result matrix calculated from c++ program
user interface