HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

COMPUTER ARCHITECTURE ASSIGNMENT REPORT

STUDENT NAME Nguyễn Lê Thảo Vy

STUDENT ID 1952536

Course ID: CO2017

Semester: 202

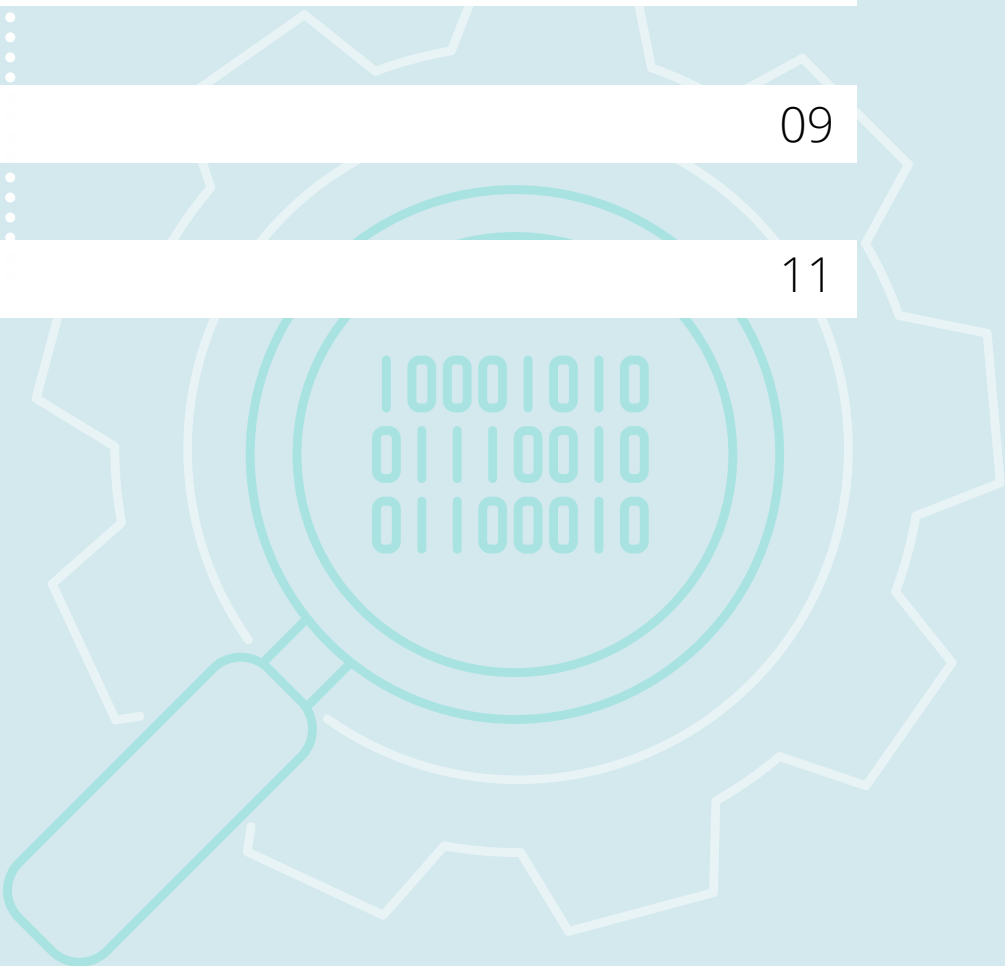
Lecturer: Dr. Phạm Quốc Cường

01

TABLE OF CONTENTS

Each question's explanation is divided to 2 part:

1/ Simplified working process
2/ Screenshots for demonstration



02

--This page is intentionally left blank--

03

QUESTION 1

Write a C/C++ program that:

- Allow user to input the dimension of two 2D matrix <A> and <B> (the matrices may be square or not).
- Read the input value from file "A.txt" and assign for the matrix <A>.
- Read the input value from file "B.txt" and assign for the matrix <B>.
- Read the input value from file "result.txt" and assign for the the matrix <result>, the height of the matrix <result> is the height of the matrix <A>, the width of the matrix <result> is the width of the matrix <B>.
- Calculate matrix <golden_result>, where <golden_result> = <A> x <B>.
- Compare the matrix <result> and <golden_result> then print the rate of difference.

Note: The number of value that are stored in "A.txt", "B.txt", "result.txt" are always equal to the size of their corresponding matrix.

The working process of the program can be simplify as below:

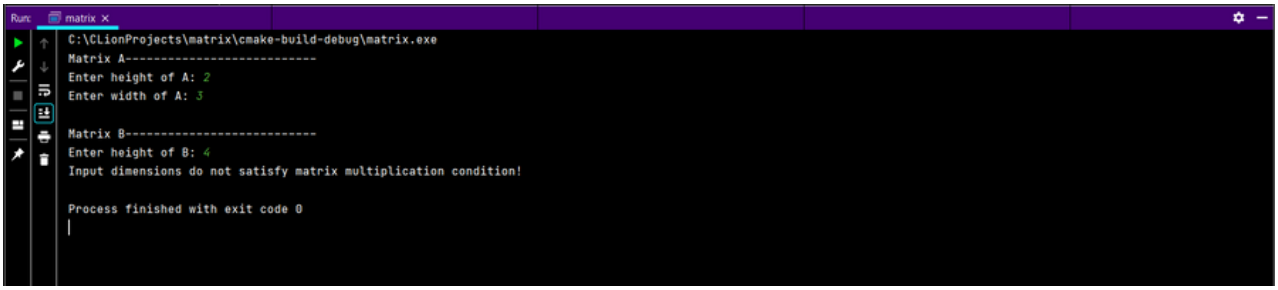
1/ Ask user to input the height and width of matrix A, then assign them to corresponding variables.
2/ Ask user to input the height of matrix B, compare it to the width of matrix A.
- If the two values do not satisfy the matrix multiplication condition, print out the error message and terminate the program.
- If the value satisfy, the program then continue to ask for input of width of matrix B.

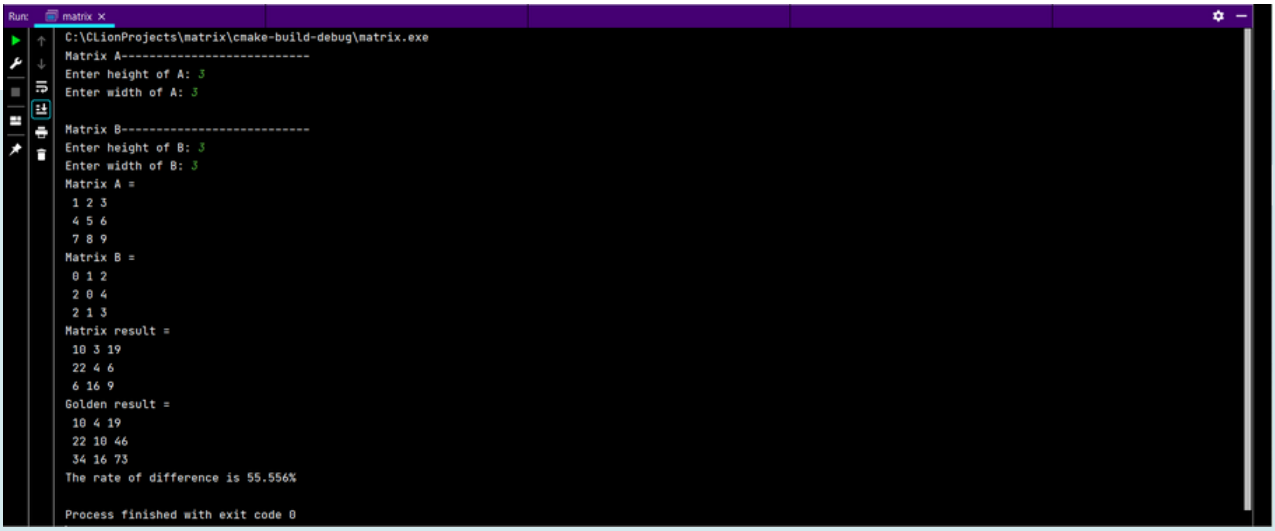
3/ Call the constructor to create two objects "A" and "B", and also create an object "result".
4/ Read the matrices A, B, and result consecutively from "A.txt", "B.txt" and "result.txt", then print out those matrices.
5/ Perform <A>x<B> and store the result into newly-created object "golden_result".
6/ Calculate the rate of difference by dividing number of elements in result/golden_result by number of different elements between them.

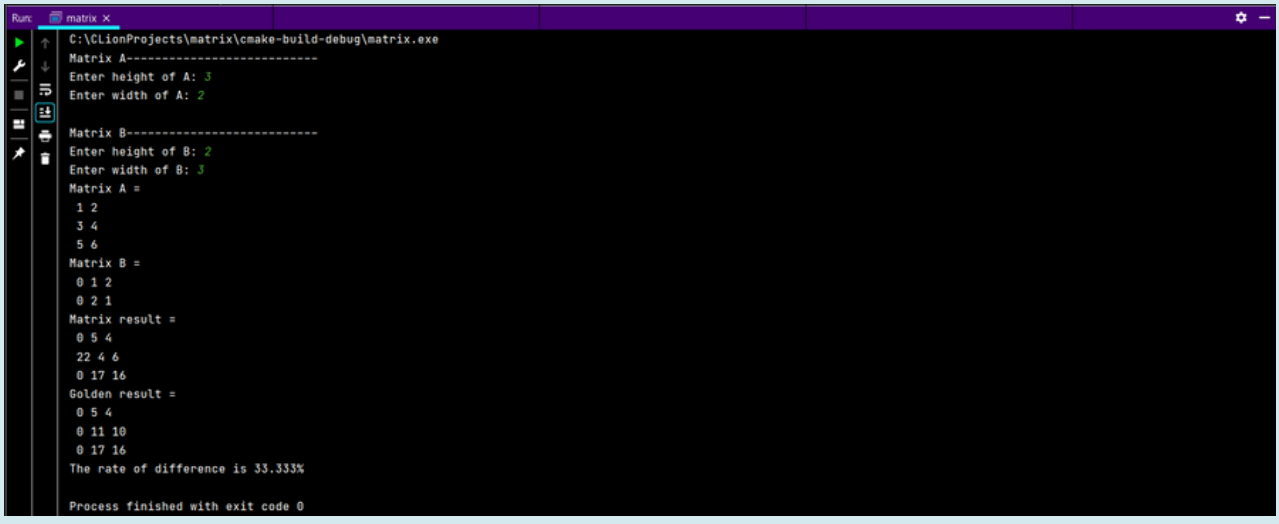
04

QUESTION 1

Some examples for demonstration







05

QUESTION 2

Write a MIPS **procedure** to perform read/write file. The input arguments satisfy the following order:

(a) Name of the file where the data are read from or written to.

(b) Argument that identify the behaviours of the procedure (read/write).

(c) The array where the data are read from (read from array and write to file) or are write to (read from file and write to array).

Function: Read/Write file.
Return value: None

The working process of the program can be simplify as below:

1/ Ask user to choose a file to read/write.

2/ Ask user to input the behavior they would like to perform, receive input and check it.

- If input = 0, perform read from file, then print out the array stored in file.
- If input = 1, perform write to file. In this case, the program will ask user to specify the array to write in, including number of elements and values of each element in array before writing given data to selected file.
- If input is different from 0 and 1, the program print out an error message and terminate itself.

Some examples for demonstration

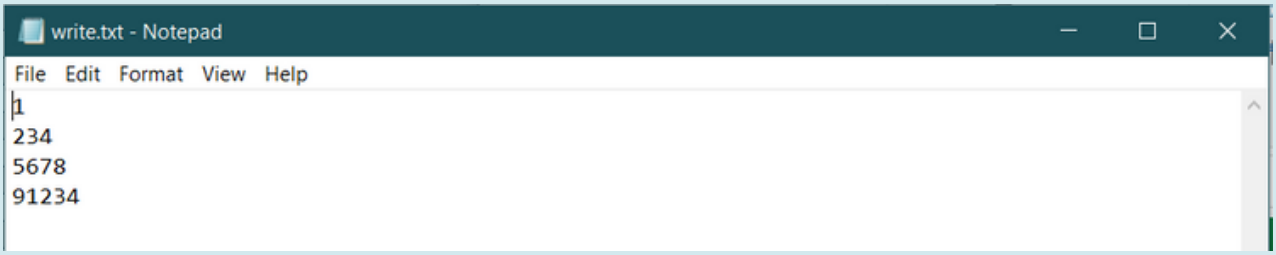
Content of file to read:

06

QUESTION 2

```
Reset: reset completed.

Enter the file name: read.txt
>> Press 0 to read selected file.
>> Press 1 to write to selected file.
Your choice is: 0

The array in file includes:
1, 22, 333, 4444
-- program is finished running --

Enter the file name: write.txt
>> Press 0 to read selected file.
>> Press 1 to write to selected file.
Your choice is: 1
Enter number of elements in array: 4
Enter a number: 1
Enter a number: 234
Enter a number: 5678
Enter a number: 91234

You can check your file now to see the result!
-- program is finished running --


Reset: reset completed.

Enter the file name: read.txt
>> Press 0 to read selected file.
>> Press 1 to write to selected file.
Your choice is: 2
Invalid input!
-- program is finished running --
```

Result in write.txt:



07

QUESTION 3

Write a MIPS **procedure** to perform dynamic allocation for an array. The input arguments satisfy the following order:

(a) The number of elements of the array.

(b) The size of each element of the array (in bytes).

Function: Dynamically allocate a space based on input requirements.
Return value 1: If total requested size exceed 65536 bytes, return an error code (-1). Otherwise, return success code (0).
Return value 2: The first address of the allocated array.

The working process of the program can be simplify as below:

1/ Ask user to input the number of elements for the array and size of each element (in bytes).

2/ Call the allocation procedure, the program will check the size needed in this procedure.

- If the size exceeds 65536 bytes, it print out an error code (-1) and terminate the program.
- If not, the procedure will assign enough memory for the array and return the base address.

3/ Print out the success code (0) and base address.

4/ User can choose between giving some inputs to make sure the allocation completed properly and skipping that part and let the program terminate at that given time.

- If user presses 1, they can continue to input some integers, and then observe the result printed out.
- If user presses 0, the program terminates. Please note that pressing any other number different from 0 and 1 would also be treated as this case.

08

QUESTION 3

Some examples for demonstration

```
Number of elements of the array: 3
Size of each element (in bytes): 4

Received success code: 0
The first address of allocated array: 0x10040000

>> You can input some integers to check if it works.
Press 0 to skip this part. Press 1 to continue: 0

-- program is finished running --


Reset: reset completed.

Number of elements of the array: 2
Size of each element (in bytes): 4

Received success code: 0
The first address of allocated array: 0x10040000

>> You can input some integers to check if it works.
Press 0 to skip this part. Press 1 to continue: 1

Enter a number: 12345

Enter a number: 6789123

Your array includes: 12345 6789123
-- program is finished running --
```

```
Number of elements of the array: 7
Size of each element (in bytes): 4

Received success code: 0
The first address of allocated array: 0x10040000

>> You can input some integers to check if it works.
Press 0 to skip this part. Press 1 to continue: 2

-- program is finished running --


Reset: reset completed.

Number of elements of the array: 2
Size of each element (in bytes): 65536

Received error code: -1
You requested more than 65536 bytes!
-- program is finished running --
```

09

QUESTION 4

Write a MIPS **procedure** to perform matrix multiplication. The input arguments satisfy the following order:

(a) The matrix A.
(b) The matrix B.
(c) The matrix result.

Function: Calculate the product of matrices A and B and store in matrix result.
Return value: If the dimension of matrices A and B does not satisfy the matrix multiplication condition, return error code (-1), else return success code (0).

The working process of the program can be simplify as below:

1/ Ask user to input the height and width of matrix A, allocate memory accordingly and get inputs for each element, then print out matrix A to console.
2/ Ask user to input the height of matrix B, compare it to the width of matrix A.
- If the two values do not satisfy the matrix multiplication condition, return error code (-1) and terminate the program.
- If the value satisfy, the program then continue to ask for input of width of matrix B, allocate memory, fill in elements and print out B.

3/ Allocate memory for the result matrix, with its height equals height of A and its width equals width of B.
4/ Perform matrix multiplication and store the result into the result matrix.
5/ Print out the result matrix and success code (0).

Some examples for demonstration

10

QUESTION 4

```
Enter number of rows for matrix A: 3
Enter number of columns for matrix A: 2
>> Enter data of matrix:
> Input data for matrix at row 0
Enter a number: 3
Enter a number: 4
> Input data for matrix at row 1
Enter a number: 5
Enter a number: 6
> Input data for matrix at row 2
Enter a number: 7
Enter a number: 8

Inserted matrix:
3 4
5 6
7 8
Enter number of rows for matrix B: 2
Enter number of columns for matrix B: 4
>> Enter data of matrix:
> Input data for matrix at row 0
Enter a number: 1
Enter a number: 2
Enter a number: 0
Enter a number: 6
> Input data for matrix at row 1
Enter a number: 8
Enter a number: 2
Enter a number: 4
Enter a number: 6

Inserted matrix:
1 2 0 6
8 2 4 6

Result of multiplication is:
35 14 16 42
53 22 24 66
71 30 32 90
Received code: 0
-- program is finished running --
```

```
Enter number of rows for matrix A: 3
Enter number of columns for matrix A: 1
>> Enter data of matrix:
> Input data for matrix at row 0
Enter a number: 1
> Input data for matrix at row 1
Enter a number: 2
> Input data for matrix at row 2
Enter a number: 3

Inserted matrix:
1
2
3
Enter number of rows for matrix B: 4
Received code: -1
Dimension error!
-- program is finished running --
```

11

QUESTION 5

Write a MIPS **program** of matrix multiplication that:

- Allow user to input the dimension of multiplier and multiplicand matrices.
- Dynamic allocate space for multiplier and multiplicand matrices.
- Dynamic allocate space for product matrix.
- Randomly generate integer value for multiplier and multiplicand matrix.
- Print the value of multiplier matrix into "A.txt" file, and multiplicand matrix into "B.txt" file.
- Calculate the product of multiplier and multiplicand matrices and store it in product matrix.
- Print the value of product matrix into "result.txt" file.
- During the program, if there are any error, print the error code then exit the program.

The working process of the program can be simplify as below:

1/ Ask user to input the height and width of matrix A, allocate memory accordingly and randomly generate values for each element, then print out matrix A to both the console and file "A.txt".

2/ Ask user to input the height of matrix B, compare it to the width of matrix A.

- If the two values do not satisfy the matrix multiplication condition, return error code (-1) and terminate the program.
- If the value satisfy, the program then continue to ask for input of width of matrix B, allocate memory, randomly fill in elements and print B on the screen and to "B.txt".

3/ Allocate memory for the result matrix, with its height equals height of A and its width equals width of B.

4/ Perform matrix multiplication and store the result into the result matrix.

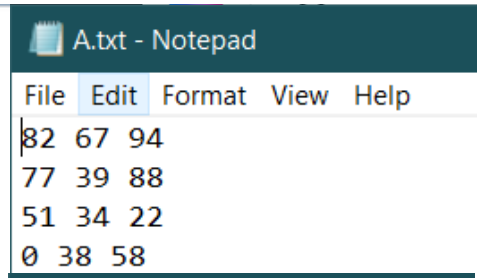
5/ Print out the result matrix and success code (0) on the screen and write the result to "result.txt".

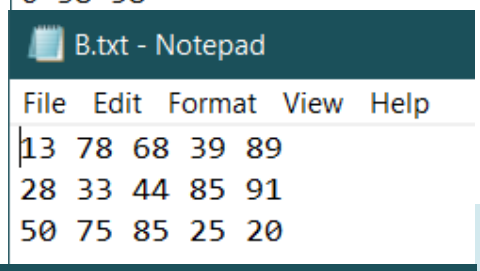
12

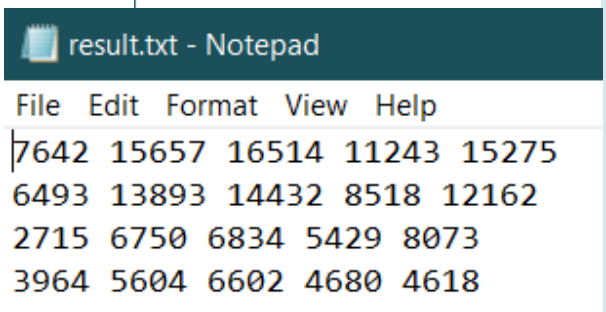
QUESTION 5

Some examples for demonstration

```
Enter number of rows for matrix A: 4
Enter number of columns for matrix A: 3

Generated matrix:
82 67 94
77 39 88
51 34 22
0 38 58
Enter number of rows for matrix B: 3
Enter number of columns for matrix B: 5

Generated matrix:
13 78 68 39 89
28 33 44 85 91
50 75 85 25 20

Result of multiplication is:
7642 15657 16514 11243 15275
6493 13893 14432 8518 12162
2715 6750 6834 5429 8073
3964 5604 6602 4680 4618

Received code: 0
-- program is finished running --
```








```
Enter number of rows for matrix A: 3
Enter number of columns for matrix A: 3

Generated matrix:
48 18 82
77 33 14
63 50 42
Enter number of rows for matrix B: 4

Received code: -1
Dimension error!
-- program is finished running --
```

--The report ends here. Thanks for your reading--