Computer Architecture

Report Assignment 2

Phạm Đình Trung - 1952512

1. Question 1:

1.1. Algorithm:

1.1.1. Get user dimension inputs by “cin”, dynamic allocate for storing matrices, declare input file stream for reading matrix entries

1.1.2. Looping through files to assign the entries to the matrices in program.

1.1.3. Close file

1.1.4. Use 3 nested “for” loops to calculate entries for the “golden_result” matrix. First loop runs from 0 -> rowA. Second loop runs from 0 -> colB. Last loop runs from 0 -> colA

1.1.5. While looping, we keep track of the differences between the entries we have just calculate “golden_result” matrix and the entries of the “result” matrix

1.1.6. Rate of difference = (number of entries which differ from “result” matrix) / (total entries of “golden” result matrix)

2. Question 2:

2.1. Idea:

2.1.1. Use the help of 2 functions: int_asciii(for write to file from array) and ascii_int (for read from file to array)

2.2. Algorithm:

2.2.1. The “read_write” function has 4 argument: Address of file name, Function, Base address of the array, Array size

2.2.2. Use “function” argument to determine what we need to do

2.2.3. For writing:

2.2.3.1. Firstly we will check if number is negative to set negative flag.

2.2.3.2. Looping through the array to pass the element of array as a argument of “int_asciii” function.

2.2.3.3. This function will extract the digit of the number we pass by modulo with 10. This can be achive by “div” to divide and “mfhi” to get the least significant digit. All the extracted digits will be plus 48 then assigned to the write buffer from high offset to low offset. After looping, we check the negative flag to add ‘-’ to the write buffer. Then we use “syscall” to write the write buffer to file.

2.2.4. For reading:

2.2.4.1. Firstly we allocate space (read buffer) which is be able to store all the value (asciii) read from file.

2.2.4.2. Then, looping through the read buffer. Every time we encounter an address store non-digit value(space or endline, minus is not counted) we will pass that address+1 to the function “ascii_int”.

2.2.4.3. Now for function “ascii_int”, it gets the argument as the address of the first asciii value in the asciii array. Firstly, this function will check if there is ‘-’ at the beginning to set negative flag. Then, looping through ascii array to calculate the final result until encounter the first non-digit value. Finally, check the negative flag to negative the result.

2.2.4.4. Back to the “read_write” function, after we call “asciii_int” function, we will store return value into the argument array

3. Question 3:

3.1. Idea: To check if the total bytes (number of element * element size) is > =0 and <= 65536. We use “sltu”. The reason is that this instruction will both check negative and > 65536. If it is negative then with “sltu”, it looks like a really big number (MSB = 1) so “sltu” sets. If it is > 65536 “sltu” sets (apparently)

3.2. Algorithm:

3.2.1. The function malloc has 2 argument: number of element, element size

3.2.2. Firstly, calculate total number of bytes need to be allocated

3.2.3. Check if the size is valid

3.2.4. If invalid then return v0 = -1

3.2.5. If valid then “syscall” and return v0 = 0

4. Question 4:

4.1. Idea:

4.1.1.The function mat_mul has 7 arguments: will be described below

4.1.2. We are working with matrices (usually use 2 dimentions array to store), we just need 1 dimension array for storing matrices. For example, 2x3 matrix will be store in an array of size 6. If we want to access the element at M[i][j], we will access the 1 dimension array at index A[(number of column)*i + j]

4.1.3. We need 7 arguments for this function (address of 3 matrices and 4 dimesions) while there are just 4 argument registers. Solution is to store the remain arguments to the stack so in the function we can get them to use.

4.2. Algorithm:

4.2.1. First check if dimensions from parameter are valid (> 0 and colA = rowB). If invalid then return v0 = -1

4.2.2. We run three nested loops as question 1 to calculate the “result” matrix. The only difference is the way to access entries. We will access the entries by using the idea in 4.1.1

5. Question 5:

5.1. Idea:

5.1.1. Random will be generated by syscall with $v0 = 42

5.1.2. Dynamic allocate will use question 3

5.1.3. Print value to file we will use question 2 but with a little change in the function:

5.1.3.1. We will modify so that the function can print the 1 dimension array as a matrix. We will pass 1 more argument to the function which is number of column. While we are looping through the array, if ((index of array+1) % number of column == 0) we will print “\n” instead of “ ” to file then continue looping

5.1.4. Calculate product matrix use question 4