**MATRIX MANIPULATOR**

Laboratory Project Report submitted for

**Computer Organisation and Architecture**

**(CSE2011)**

Submitted by

|  |  |
| --- | --- |
| **GAURAV KUMAR**  **Registration No.: 1541017302** | **ANKITA SAHU**  **Registration No.: 1541017303** |
| **SHREYA**  **Registration No.: 1541017304** | **SAURAV KUMAR SINGH**  **Registration No.: 1541017305** |

**(CSIT ’D’ 4th SEMESTER)**



Department of Computer Science and Engineering

Institute of Technical Education and Research

(Faculty of Engineering)

**SIKSHA ‘O’ ANUSANDHAN UNIVERSITY**

**Bubaneswar, Odisha, India**

(JAN-May, 2017)

# Declaration

We, the undersigned students of B. Tech. of Computer Science and Engineering Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled “**(MATRIX MANIPULATOR)**” submitted to **Siksha ‘O’ Anusandhan University, Bhubaneswar** for the partial fulfillment of the subject **Computer Organisation and Architecture (CSE2011)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

|  |  |
| --- | --- |
| **GAURAV KUMAR**  **Registration No.: 1541017302** | **ANKITA SAHU**  **Registration No.: 1541017303** |
| **SHREYA**  **Registration No.: 1541017304** | **SAURAV KUMAR SINGH**  **Registration No.: 1541017305** |

**DATE: 21.04.17**

**PLACE:ITER ,SOA university**

**BHUBANESWAR**

# Abstract

In computation involving matrices it is frequently necessary to interchange or rearrange rows or columns of a matrix. If the work is being done longhand or with a desk calculator , it is desirable to be able to perform the rearrangement without having to erase or rewrite numbers. In mathematics, matrix addition or subtraction, multiplication or the matrix product or any other arithmetic operation is a binary operation that produces a matrix from two matrices.

The matrix manipulator was devised for the use in the Bureau’s statistical Engineering Laboratory for calculation with incidence matrices, i.e., matrices whose element are all 0’s or 1’s. So, through this project we will show how the matrix manipulation is taking place using MIPS programming language. MIPS is a reduced instruction set computer set architecture developed by MIPS Technologies. The early MIPS architecture were 32 bit with 64 bit versions added later. Also the project is compiled up to show Matrix addition of two different matrices, subtraction of two different matrices ,also scaling up with a constant multiplication and transpose of matrix in respect of the fact that in both the matrices ,the required number of rows and columns present.

# Contents

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial No.** | **Chapter No.** | **Title of the Chapter** | **Page No.** |
|  | 1 | Introduction | 5 |
|  | 2 | Problem Statement | 6 |
|  | 3 | Brief Description | 7 |
|  | 4 | Steps of the Algorithms/Flow Diagram | 8 |
|  | 5 | Source Code using MIPS | 10 |
|  | 6 | Testing | 22 |
|  | 7 | Conclusion | 25 |
|  |  | References | 26 |

# 

# 1. Introduction

This project has been done keeping in mind all the basic concepts taught in Computer Organization and Architecture. This project deals with the matrix manipulation of two matrices. It is well known that matrices can be used in several key areas of science to provide meaning to data. Often these matrices can be manipulated to provide a solution based on the important properties of the matrices like Matrix addition, subtraction of two different matrices ,also scaling up with a constant multiplication and transpose of matrix in respect of the fact that in both the matrices ,the required number of rows and columns present.

The user is prompted to enter two 3X3 matrixes. The user is also asked to enter the choice of operation to be done with the inputted matrices i.e. a)Matrix Addition, b)Matrix Subtraction, c)Scaling, and d)Transpose e) Multiplication using user defined functions, and the output is displayed. If user enters invalid option then error message is displayed.

# 2. Problem Statement

To develop matrix operation calculator using MIPS . It consists of basic operation like addition, subtraction, scaling and transpose and multiplication. Addition or subtraction is accomplished by adding or subtracting corresponding elements. Matrix addition is used to add up the respective elements present in two different matrices and store it into another matrix to show the addition property of the matrix. Similarly, Matrix subtraction is to subtract the respective elements present in two different matrices and store it in another matrix to show the subtraction property of the matrix. Scaling is the transformation, when applied to an object multiplies each of the local coordinates. Here Scaling is used to scaled up/down the matrix with a constant using scalar multiplication. Transpose of matrix is that in which the all the rows of a given matrix is transformed into columns and vice-versa. Multiplication of matrices is also done using MIPS code by rows of first matrix with the columns of second matrix.

# 3. Brief Description

As already mentioned in the introduction, matrix manipulator system will be required to carry out various operations and provide meaning to these combined structures. This section will outline the various features, implementations and user interactions provided by the system. During this project the basic matrix operations such as matrix, addition, subtraction, scaling and transpose. In Matrix Addition and subtraction, we assume the two matrices given in the data segment. All we have to do is to start with the base address add the respective elements of the two matrices , increment the address by four and do the same till the length is reached. While approaching for the Scaling of matrices there are many prospects to show scaling. We chose the one with scalar multiplication of the matrix with a constant. The next operation assigned to us is transpose in which we just have to reverse the rows with the columns and the columns with the rows simultaneously. The last operation is multiplication of the matrix. If the user enters invalid option then appropriate message is displayed.

# 4. Steps of the Algorithms and Flow Diagram

1) Input the two matrixes each of size 3\*3 from the user in the console using various syscall services.

2) Prompt the user to enter the choice of the operation and store it in $s7 register.

3)

A>If the content of $s7 register=1 then perform addition of the two matrices (using addition function).

B>If the content of $s7 register=2 then perform subtraction of the two matrices (using subtraction function).

C> If the content of $s7 register=3 then perform transpose of two matrices (using transpose function) .

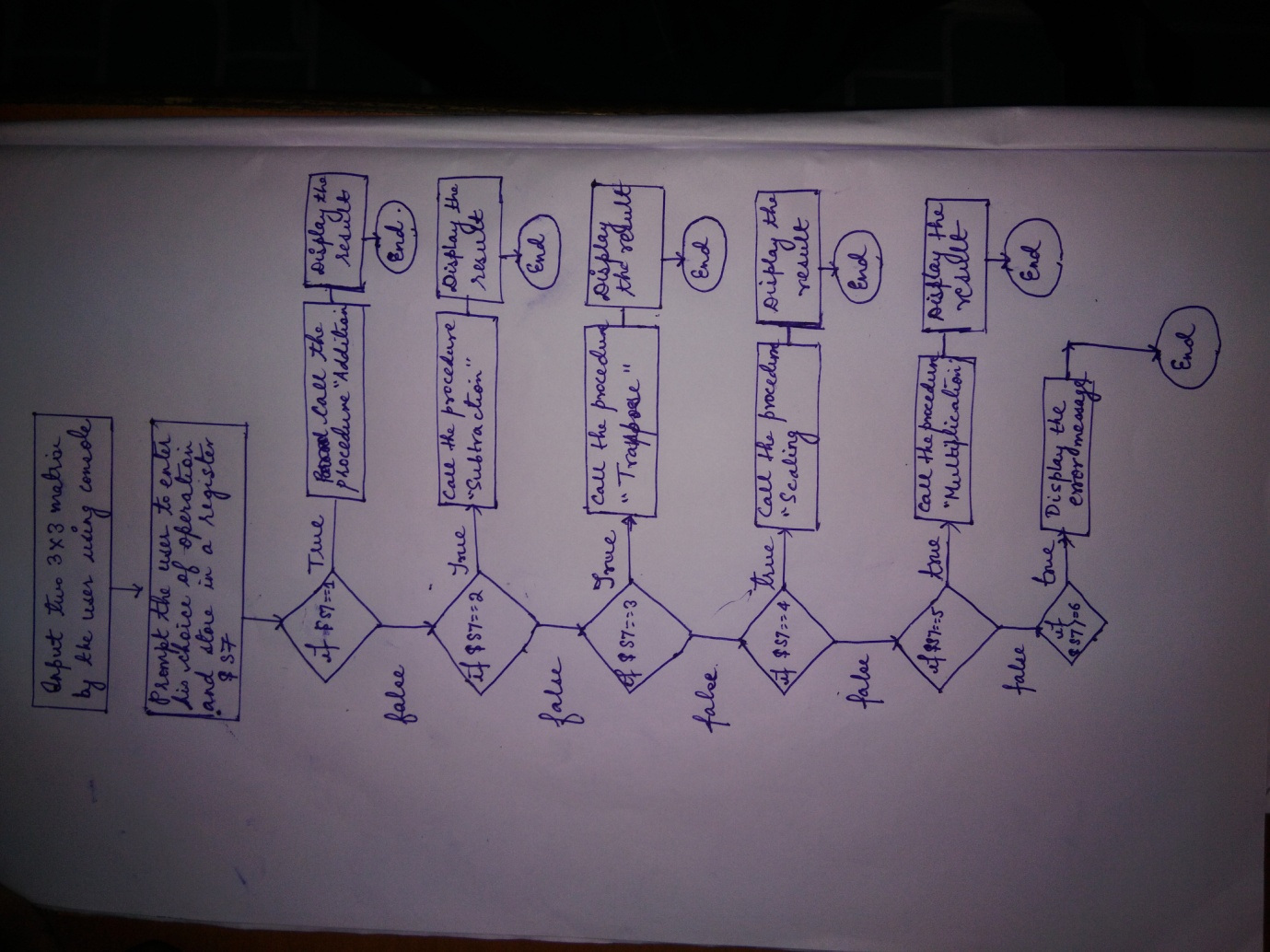
D>If the content of $s7 register=4

then perform scaling for that take a scaling element.

E>If the content of $s7 register=5

then perform multiplication (using multiplication function).

F>For any other choice, display the error message and stop.



# 5. Source Code using MIPS

.data

num: .asciiz "Enter a number to be scaled with the matrix:> "

matrix1: .word 1 2 3 4 5 6 7 8 9

matrix2: .word 1 2 3 4 5 6 7 8 9

result: .word 0 0 0 0 0 0 0 0 0

row\_index: .word 0

column\_index: .word 0

length\_of\_row: .word 3

length\_of\_column: .word 3

scaling\_ele: .word 0

spac: .asciiz"\t"

entr: .asciiz"\n"

mat1: .asciiz"\n enter 1st matrix"

mat2: .asciiz"\n enter 2nd matrix"

msg2: .asciiz"\t\t THIS IS A MATRIX MANIPLUATOR\n"

msg1: .asciiz "\tCHOICES:-\n1-Addition\n2-Subtraction\n3-Transpose\n4-Scaling\n5-multiplication\n6-stop\n"

msg3: .asciiz"\t\THANK YOU\n"

n: .asciiz "ENTER YOUR CHOICE:> "

error: .asciiz "WRONG CHOICE\n"

add\_res: .asciiz"\nADDITION RESULT IS:-\n"

sub\_res: .asciiz"\nSUBTRACTION RESULT IS:-\n"

tra\_res: .asciiz"\nTRANSPOSE RESULT IS:-\n"

sca\_res: .asciiz"\nSCALING RESULT IS:-\n"

mul\_res: .asciiz"\nMULTIPLICATION RESULT IS:-\n"

.text

main:

li $v0,4

la $a0,msg2

syscall

li $v0,4

la $a0,msg1

syscall

la $a2,matrix1

la $a1,matrix2

INPUT:

li $v0,4

la $a0,n

syscall

li $v0,5

syscall

move $s7,$v0

1

li $s3,0

bgt $s7,$s3,first

li $v0,4

la $a0,error

syscall

j INPUT

first:

add $s3,$s3,1

bne $s7,$s3,second

jal addition

j INPUT

second:

add $s3,$s3,1

bne $s7,$s3,third

jal subtraction

j INPUT

third:

add $s3,$s3,1

bne $s7,$s3,fourth

jal transpose

j INPUT

fourth:

add $s3,$s3,1

bne $s7,$s3,fifth

li $v0,4

la $a0,num

syscall

li $v0,5

syscall

move $t0,$v0

sw $t0,scaling\_ele

jal scaling

j INPUT

fifth:

add $s3,$s3,1

bne $s7,$s3,sixth

jal multiplication

j INPUT

sixth:

addi $s3,$s3,1

bne $s7,$s3,seventh

j exit

seventh:

li $v0,4

la $a0,error

syscall

j INPUT

exit:

li $v0,4

la $a0,msg3

syscall

li $v0,10

syscall

li $v0,4

la $a0,mat1

syscall

#------------------------------------------------------------

#Taking Input for matrix-1

MATRIX1:

li $t0,0 #row\_indx

li $t2,3 #No\_of\_rows

li $t3,3 #No\_of\_columns

la $t4, matrix1 #starting addrs of array

ILOOP:

li $t1,0 #col\_indx

JLOOP:

mul $t5, $t0, 3 #row\_size size of each row

add $t5, $t5, $t1 #t5 = i \* no\_of\_row + j

mul $t5, $t5, 4 #t5 \* word\_size

li $v0,5

syscall

add $t5,$t5,$t4

sw $v0,($t5) #store a[i][j] = v

addi $t1, $t1, 1 #j++

blt $t1, $t3, JLOOP

addi $t0, $t0, 1 #i++

blt $t0,$t2, ILOOP

li $v0,4

la $a0,mat2

syscall

#Taking Input for matrix-1

MATRIX2:

li $t0,0 #row\_indx

li $t2,3 #No\_of\_rows

li $t3,3 #No\_of\_columns

la $t4, matrix2 #starting addrs of array

I2LOOP:

li $t1,0 #col\_indx

J2LOOP:

mul $t5, $t0, 3 #row\_size size of each row

add $t5, $t5, $t1 #t5 = i \* no\_of\_row + j

mul $t5, $t5, 4 #t5 \* word\_size

li $v0,5

syscall

add $t5,$t5,$t4

sw $v0,($t5) #store a[i][j] = v

addi $t1, $t1, 1 #j++

blt $t1, $t3, J2LOOP

addi $t0, $t0, 1 #i++

blt $t0, $t2, I2LOOP

#------------------------------------------------

.end main

.ent addition

.globl addition

addition:

lw $t1,column\_index

lw $t0,row\_index

lw $t2,length\_of\_row

lw $t3,length\_of\_column

la $s5,spac

la $s6,entr

#ADDITION

li $v0,4

la $a0,add\_res

syscall

loop1:

blt $t0,$t2,loop

j exit1

loop:

mul $t6,$t2,$t0

add $t6,$t6,$t1

sll $t6,$t6,2

add $t7,$t6,$a2

add $t8,$t6,$a1

lw $t9,($t7)

lw $s0,($t8)

add $t9,$t9,$s0

li $v0,1

move $a0,$t9

syscall

li $v0,4

la $a0,spac

syscall

addi $t1,$t1,1

blt $t1,$t3,loop

addi $t0,$t0,1

add $t1,$zero,$zero

li $v0,4

la $a0,entr

syscall

j loop1

exit1:

jr $ra

.end addition

.ent subtraction

.globl subtraction

subtraction:

li $v0,4

la $a0,sub\_res

syscall

#SUBTRACTION

lw $t1,column\_index

lw $t0,row\_index

lw $t2,length\_of\_row

lw $t3,length\_of\_column

la $s5,spac

la $s6,entr

li $v0,4

la $a0,entr

syscall

loop2:

blt $t0,$t2,loop3

j exit2

loop3:

mul $t6,$t2,$t0

add $t6,$t6,$t1

sll $t6,$t6,2

add $t7,$t6,$a2

add $t8,$t6,$a1

lw $t9,($t7)

lw $s0,($t8)

sub $t9,$t9,$s0

li $v0,1

move $a0,$t9

syscall

li $v0,4

la $a0,spac

syscall

addi $t1,$t1,1

blt $t1,$t3,loop3

addi $t0,$t0,1

add $t1,$zero,$zero

li $v0,4

la $a0,entr

syscall

j loop2

exit2:

jr $ra

.end subtraction

.ent transpose

.globl transpose

transpose:

li $v0,4

la $a0,tra\_res

syscall

#Transpose

lw $t1,column\_index

lw $t0,row\_index

lw $t2,length\_of\_row

lw $t3,length\_of\_column

la $s5,spac

la $s6,entr

li $v0,4

la $a0,entr

la $s2,result

syscall

loop4:

blt $t0,$t2,loop5

j exit3

loop5:

mul $t6,$t2,$t0

add $t6,$t6,$t1

sll $t6,$t6,2

add $t7,$t6,$a2

add $t4,$t6,$s2

lw $t8,($t7)

#sw $t8,($t4)

ble $t0,$t1,swap

j printing

swap:

mul $t9,$t2,$t1

add $t9,$t9,$t0

sll $t9,$t9,2

add $s0,$t9,$a2

add $t5,$t9,$s2

lw $s1,($s0)

sw $t8,($t5)

sw $s1,($t4)

#lw $a0,($t7)

#move $a0,$t8

#li $v0,1

#syscall

#j printing1

printing:

lw $a0,($t4)

#move $a0,$s1

li $v0,1

syscall

#printing1:

li $v0,4

la $a0,spac

syscall

addi $t1,$t1,1

blt $t1,$t3,loop5

addi $t0,$t0,1

add $t1,$zero,$zero

li $v0,4

la $a0,entr

syscall

j loop4

exit3:

jr $ra

.end transpose

.ent scaling

.globl scaling

scaling:

li $v0,4

la $a0,sca\_res

syscall

#Scaling

lw $t1,column\_index

lw $t0,row\_index

lw $t2,length\_of\_row

lw $t3,length\_of\_column

la $s5,spac

lw $s4,scaling\_ele

la $s6,entr

li $v0,4

la $a0,entr

syscall

loop6:

blt $t0,$t2,loop7

j exit4

loop7:

mul $t6,$t2,$t0

add $t6,$t6,$t1

sll $t6,$t6,2

add $t7,$t6,$a2

lw $t8,($t7)

mul $t9,$t8,$s4

#sw $t9,($t7)

#lw $a0,($t7)

move $a0,$t9

li $v0,1

syscall

li $v0,4

la $a0,spac

syscall

addi $t1,$t1,1

blt $t1,$t3,loop7

addi $t0,$t0,1

add $t1,$zero,$zero

li $v0,4

la $a0,entr

syscall

j loop6

exit4:

jr $ra

.end scaling

.ent multiplication

.globl multiplication

multiplication:

li $v0,4

la $a0,mul\_res

syscall

#-----------------------------------------------------------------------------

#Multiplication

li $t0, 0 #i

la $t3, matrix1

la $t4, matrix2

la $t5, result

Loop1:

li $t1, 0 #j

Loop2:

li $t2, 0 #k

move $t9, $zero #temp var to store sum

Loop3:

mul $t6, $t0, 3 # I \* row\_size

add $t6, $t6, $t2 #i \* rowsize + k

#move $a0, $t6

#li $v0, 4

#syscall

mul $t6, $t6, 4

add $t6, $t6, $t3

lw $t6, ($t6) #t6=a[i][k]

mul $t7, $t2, 3 #k \* rowSize

add $t7, $t7, $t1 #k\*row + j

mul $t7, $t7, 4

add $t7, $t7, $t4

lw $t7, ($t7) #t7=b[k][j]

#c[i][j]+=a[i][k] \* b[k][j]

mul $t6, $t7, $t6

add $t9, $t9, $t6

add $t2, $t2, 1

blt $t2, 3, Loop3 #k < 3

#sw $t9, ($t8) #c[i][j]=t9

move $a0,$t9

li $v0,1

syscall

li $v0,4

la $a0,spac

syscall

add $t1, $t1, 1

blt $t1, 3, Loop2 #j < 3

li $v0,4

la $a0,entr

syscall

add $t0, $t0, 1

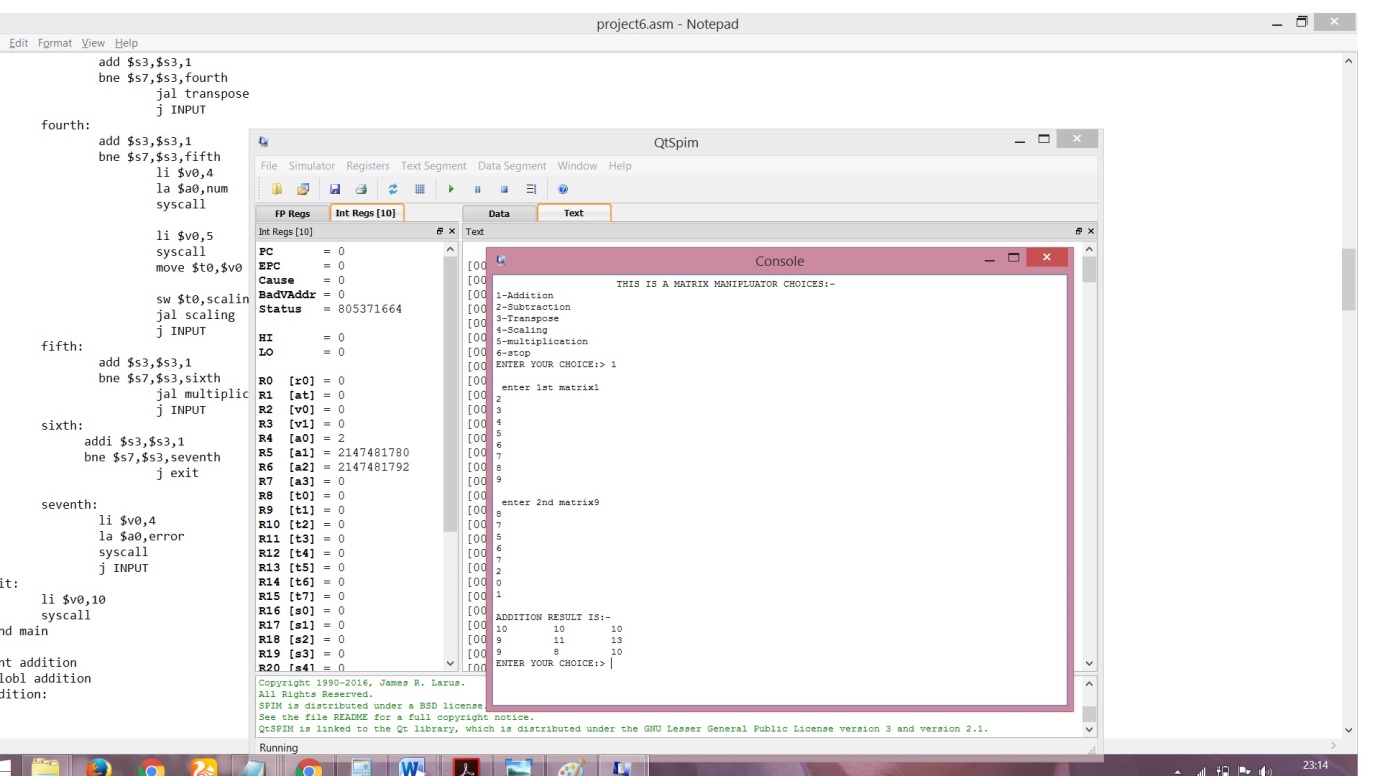
blt $t0, 3, Loop1 #i < 3

jr $ra

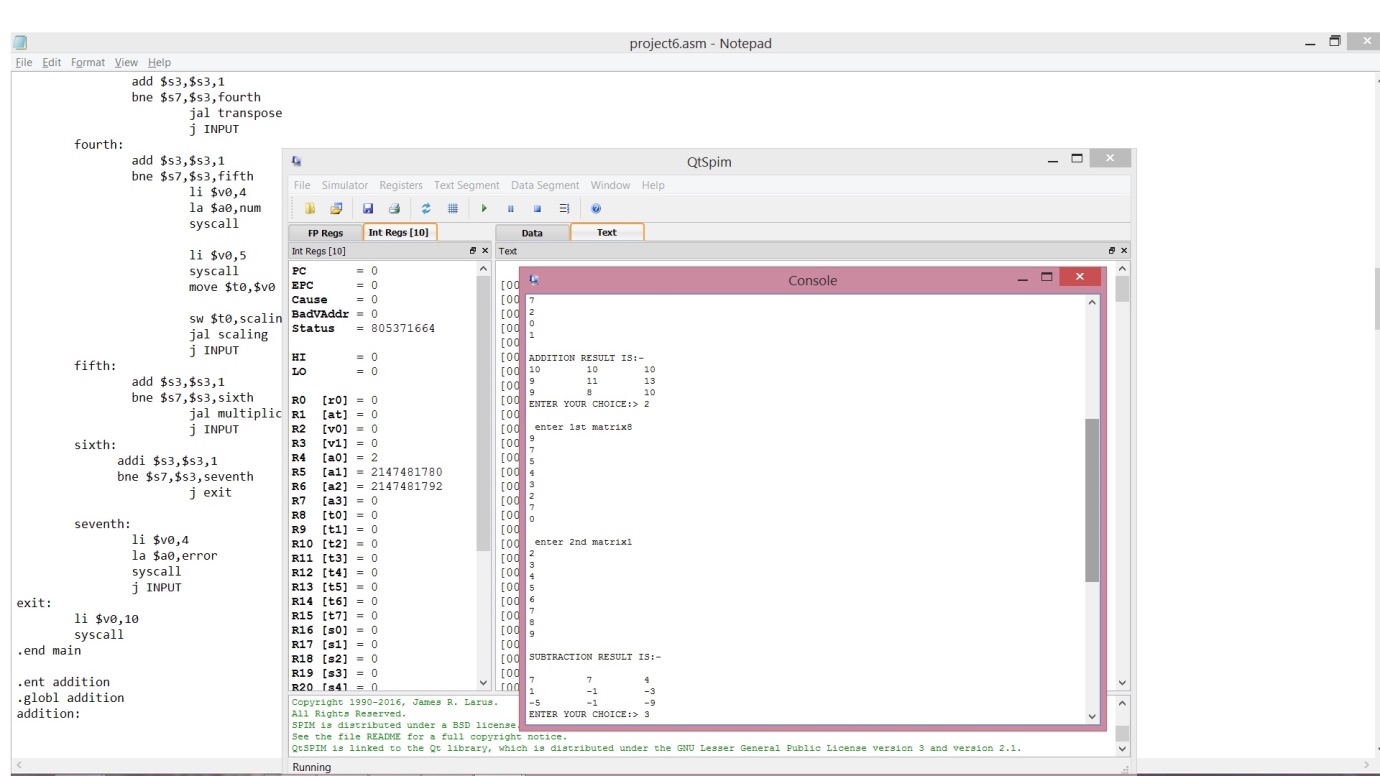
.end multiplication

**6. Testing**

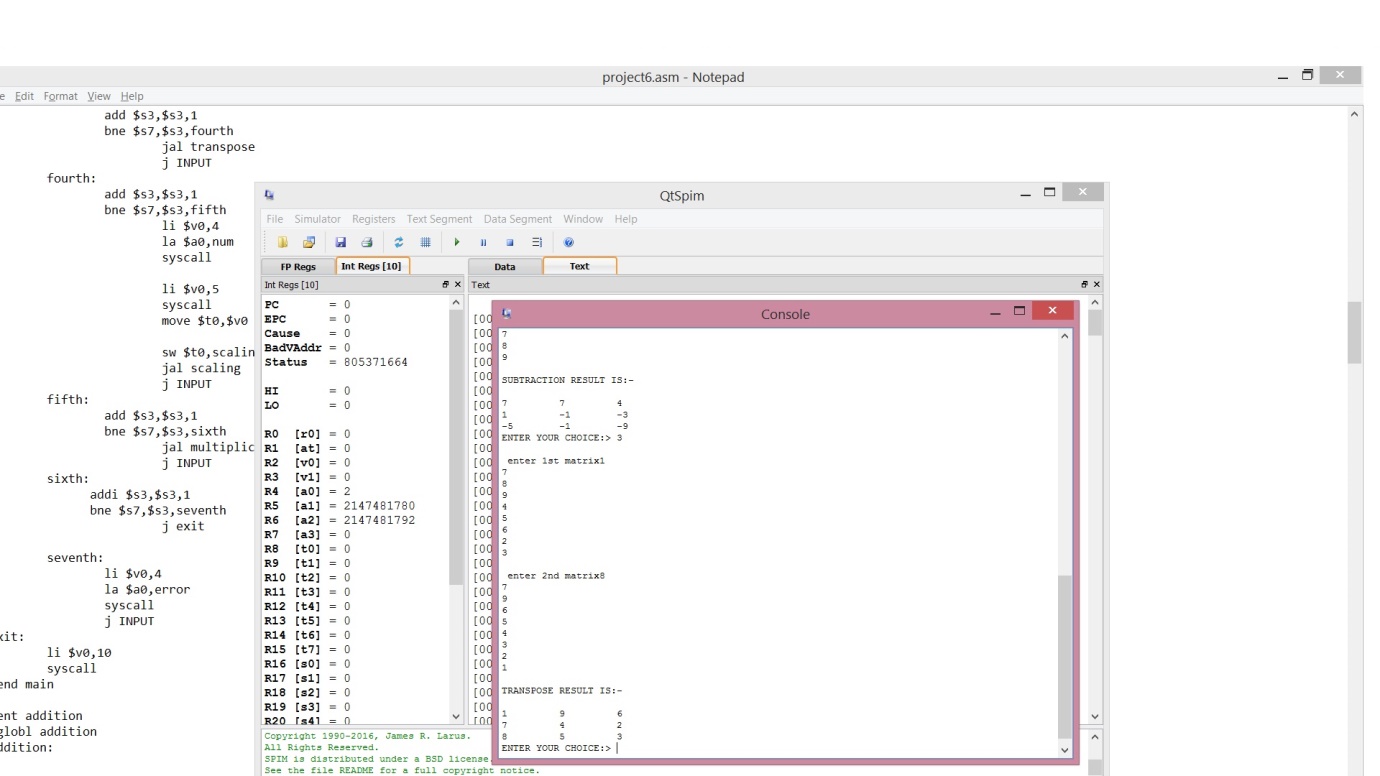
**Output:1**

****

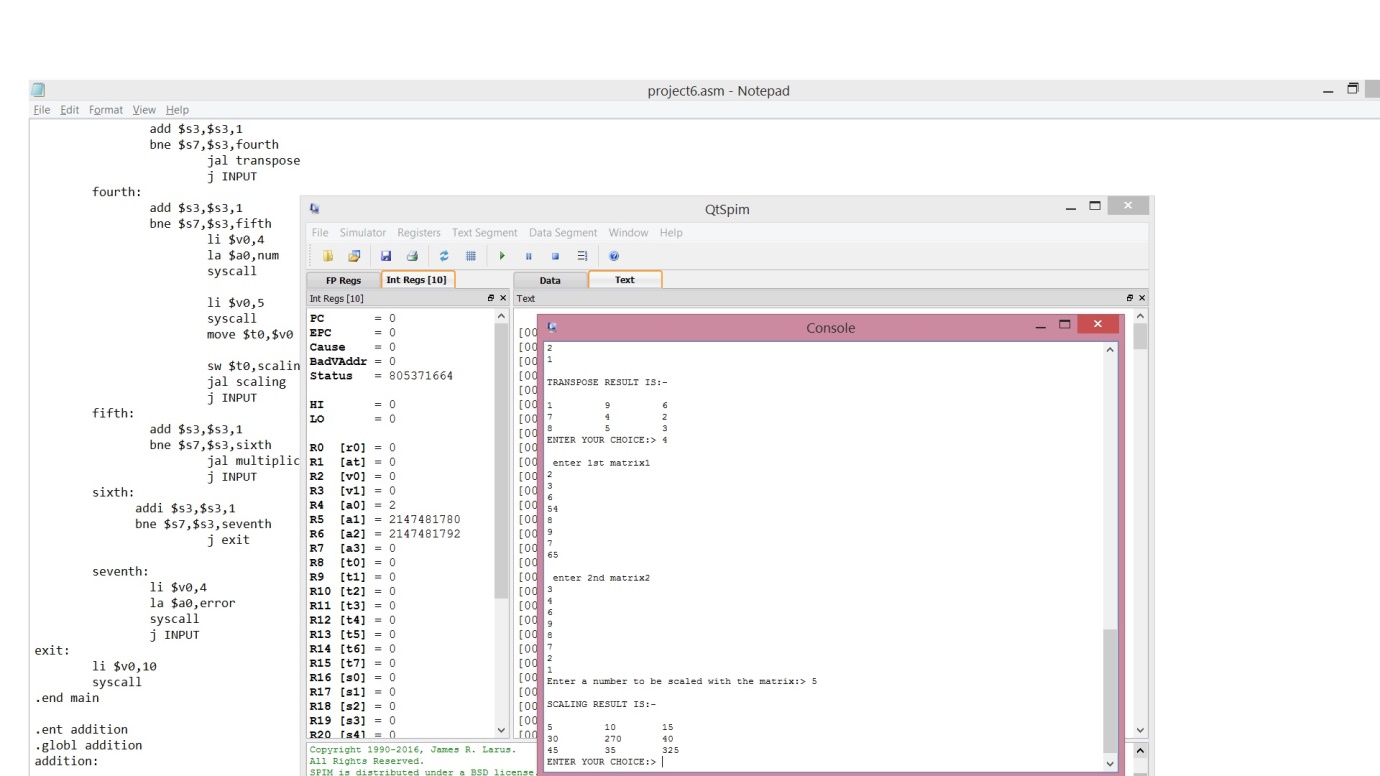
**Output:2**

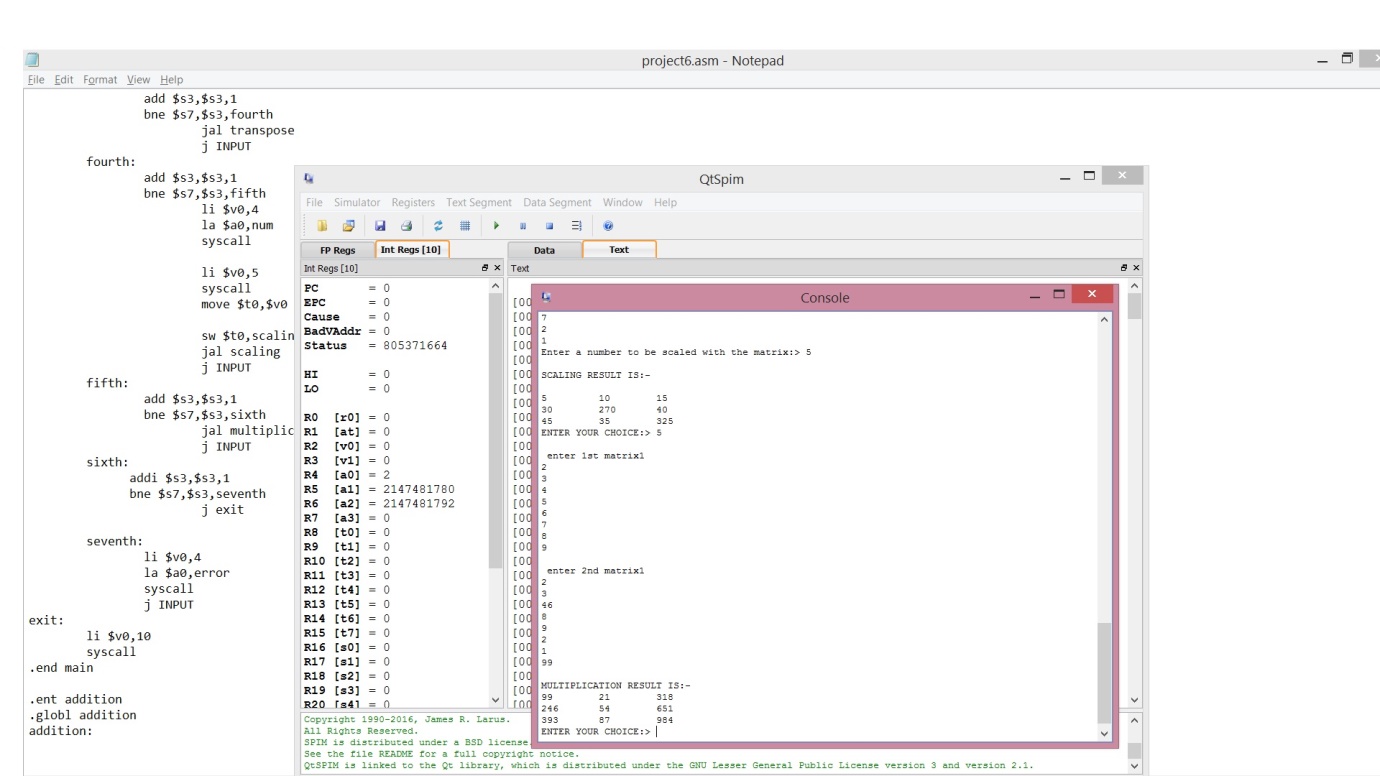
****

**Output:3**

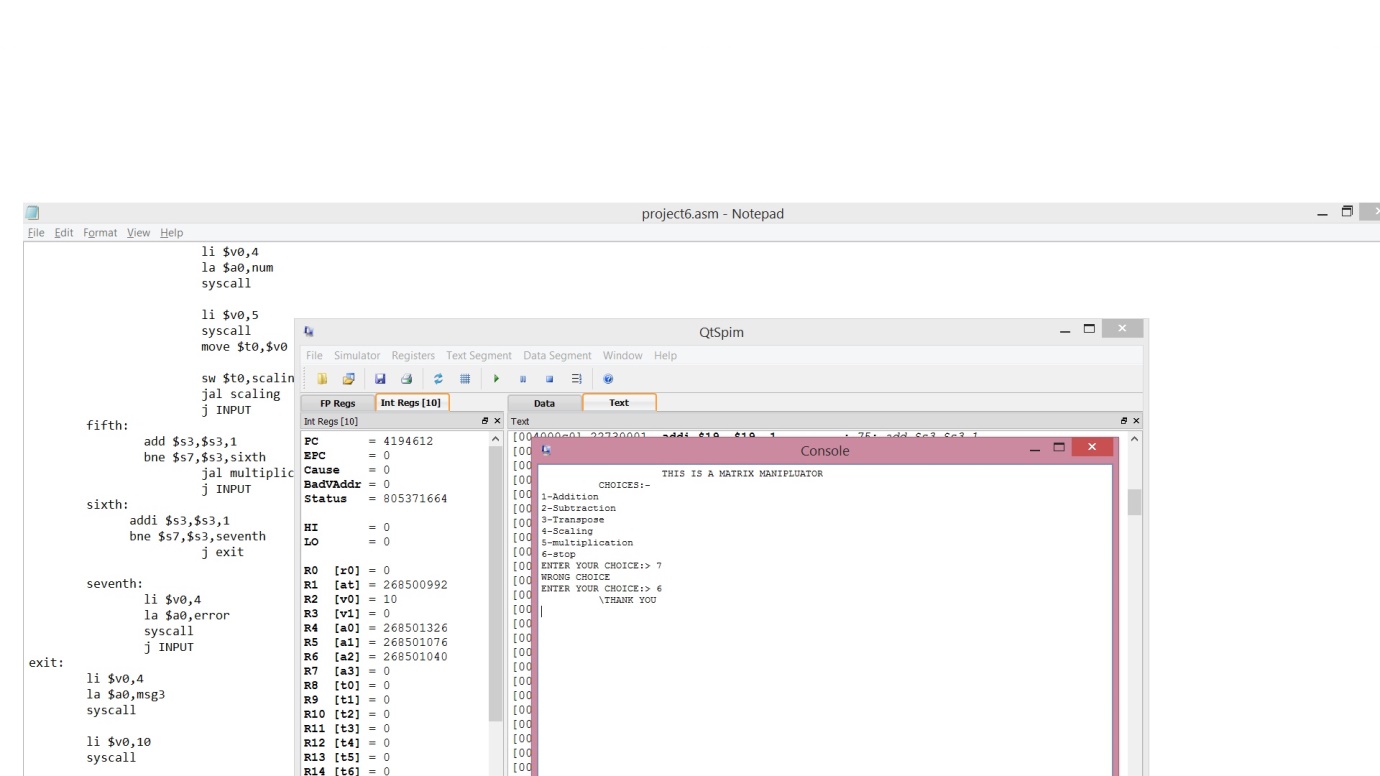
****

**Output:4**

****

**Output:5**

**Output:6**

****

# 7. Conclusion

In order to simplify matrix manipulation, we designed a MIPS code for Matrix Manipulator to test and implement different arithmetic operations like Matrix Addition, Matrix Subtraction, matrix scaling and matrix transpose and matrix multiplication. This project will comprise of all the arithmetic operations of the matrix assigned to us, the fact that a set of array is assumed in form of matrix and the operations were done.

The limitations that came across while designing the project was

* 1. The matrices must be a square matrix
  2. For scaling and transposing two matrixes is needed to taken as a input.

# References

1 .COMPUTER ORGANIZATION AND DESIGN

By David A Patterson and John L Hennessy

2. <https://www.jstor.org/stable/3029500>

3. stackoverflow.com/questions/30232649/matrix-multiplication-in-mips-array-looping