##

## lab5.asm is a program that reads a text file as a matrix and adds matrices

##

#########################################

# #

# text segment #

# #

#########################################

.text

.globl main

main: #execution starts here

li $v0, 30

syscall

move $s5, $a0

la $t0, size

lw $s0, 0($t0) # $s0 = N

mul $a0, $s0, $s0 # N^2

sll $a0, $a0, 2 # 4xN^2

li $v0, 9 #HEAP

syscall

move $s2, $v0 #store the address of resulting matrix

# Store the matrices read from the files

li $v0, 13

la $a0, mat1

li $a1, 0

li $a2, 0

syscall

move $t1, $v0

#read

li $v0, 14

move $a0, $t1

la $a1, stringBuff

li $a2, 160000000

syscall

move $s1, $v0 #$s1 = no of read chars

#close

li $v0, 16

move $a0, $t1 #restore file desc

syscall

#read buffer for mat1

la $a0, stringBuff

move $a1, $s1

move $a2, $s0

jal read\_buffer

move $s3, $v0 #get the adress of the matrix1

# Store the matrix2 read from the file

li $v0, 13

la $a0, mat2

li $a1, 0

li $a2, 0

syscall

move $t1, $v0

#read

li $v0, 14

move $a0, $t1

la $a1, stringBuff

li $a2, 160000000

syscall

move $s1, $v0 #$s1 = no of read chars

#close

li $v0, 16

move $a0, $t1 #restore file desc

syscall

#read buffer for mat2

la $a0, stringBuff

move $a1, $s1

move $a2, $s0

jal read\_buffer

move $s4, $v0 #get the adress of the matrix2

li $v0, 30

syscall

move $s6, $a0

sub $a0, $s6, $s5

li $v0, 1

syscall

li $v0, 4

la $a0, spce

syscall

#matrix addition

move $a0, $s3## $a0 = base address of matrix B

move $a1, $s4## $a1 = base address of matrix C

move $a2, $s2## $a2 = base address of matrix A

move $a3, $s0## $a3 = N

jal matrix\_addition

li $v0, 30

syscall

sub $a0, $a0, $s6

li $v0, 1

syscall

li $v0, 10

syscall # bye bye

########## READ BUFFER ##########

## $a0 = string buffer address

## $a1 = no of chars

## $a2 = N

## $v0 = adress of matrix

read\_buffer:

addi $sp, $sp, -32

sw $ra, 0($sp)

sw $s0, 4($sp)

sw $s1, 8($sp)

sw $s2, 12($sp)

sw $s3, 16($sp)

sw $s4, 20($sp)

sw $s7, 24($sp)

sw $t9, 28($sp)

move $t9, $a0

move $t0, $a0 #index1

move $s1, $a2 # $s1 = N

move $s7, $a1

add $s7, $s7, $a0

## allocate heap space

la $t3, size

lw $t4, 0($t3) # $s0 = N

mul $a0, $t4, $t4 # N^2

sll $a0, $a0, 2 # 4xN^2

li $v0, 9 #HEAP

syscall

move $s0, $v0 #store the address of matrix

move $s4, $s0

j test\_rb

read\_loop:

lb $t3, 0($t0)

sle $t4, $t3, 57

sge $t5, $t3, 48

and $t4, $t5, $t4

beq $t4, $0, increment

inner\_loop:

addi $t1, $t0,1 #index2 = index1+1

j inner\_test

increment\_i2:

addi $t1, $t1, 1

inner\_test:

lb $t6, 0($t1)

sle $t7, $t6, 57

sge $t8, $t6, 48

and $t7, $t8, $t7

bne $t7, $0, increment\_i2

addi $a1, $t0,0

addi $a2, $t1,-1

jal string\_to\_int

move $s3, $v0

sw $s3, 0($s4) # matrix[i] = converted int

addi $t0, $t1,0 #index1 = index2 + 1 %%%%%

addi $s4, $s4, 4

increment:

addi $t0, $t0, 1

test\_rb: bne $t0, $s7, read\_loop

move $v0, $s0 #return base adress of created matrix

lw $t9, 28($sp)

lw $s7, 24($sp)

lw $s4, 20($sp)

lw $s3, 16($sp)

lw $s2, 12($sp)

lw $s1, 8($sp)

lw $s0, 4($sp)

lw $ra, 0($sp)

addi $sp, $sp, 32

jr $ra

######### STRING TO INT #########

## $a1 = string index1

## $a2 = string index2

## $v0 = converted integer

string\_to\_int:

addi $sp, $sp, -16

sw $t1, 12($sp)

sw $t2, 8($sp)

sw $t3, 4($sp)

sw $t0, 0($sp)

move $v0, $0

addi $t1, $a2, 0 # i = index2

addi $t2, $0, 1 # times 10^n register

j test

loopConvert:

lb $t3, 0($t1)

subi $t3, $t3, 48 # char to int in ascii values

mul $t3, $t2, $t3 # decimal conversion for the digit

add $v0, $v0, $t3

mul $t2, $t2, 10

addi $t1, $t1, -1

test: bge $t1, $a1, loopConvert

end:

lw $t1, 12($sp)

lw $t2, 8($sp)

lw $t3, 4($sp)

lw $t0, 0($sp)

addi $sp, $sp, 16

jr $ra

######## MATRIX ADDITION ########

## $a0 = base address of matrix B

## $a1 = base address of matrix C

## $a2 = base address of matrix A

## $a3 = N

matrix\_addition:

addi $t0, $0, 0 #i = 0

for\_out\_add:

beq $t0, $a3, for\_out\_end\_add #i! = N

addi $t1, $0, 0 #j = 0

for\_in\_add:

beq $t1, $a3, for\_in\_end\_add #j! = N

#calculate addresses

mul $t7, $t0, $a3 #i\*N

add $t7, $t7, $t1 #i\*N+j

sll $t7, $t7, 2 #(i\*N+j)\*4

add $t3, $t7, $a0 #(base address of matrix B) + offset

add $t4, $t7, $a1 #(base address of matrix C) + offset

add $t5, $t7, $a2 #(base address of resultant matrix A) + offset

#sum

lw $t9, ($t5) #load matrix A

lw $t8, ($t3) #load matrix B

lw $t6, ($t4) #load matrix C

add $t9, $t8, $t6 #add

sw $t9, ($t5)

addi $t1, $t1, 1

j for\_in\_add

for\_in\_end\_add:

addi $t0, $t0, 1

j for\_out\_add

for\_out\_end\_add:

jr $ra

######## MATRIX ADDITION LOOP INDTERCHANGE ########

## $a0 = base address of matrix B

## $a1 = base address of matrix C

## $a2 = base address of matrix A

## $a3 = N

matrix\_addition\_intr:

addi $t1, $0, 0 #j = 0

for\_out\_add\_intr:

beq $t1, $a3, for\_out\_end\_add\_intr #j! = N

addi $t0, $0, 0 #i = 0

for\_in\_add\_intr:

beq $t0, $a3, for\_in\_end\_add\_intr #i! = N

#calculate addresses

mul $t7, $t0, $a3 #i\*N

add $t7, $t7, $t1 #i\*N+j

sll $t7, $t7, 2 #(i\*N+j)\*4

add $t3, $t7, $a0 #(base address of matrix B) + offset

add $t4, $t7, $a1 #(base address of matrix C) + offset

add $t5, $t7, $a2 #(base address of resultant matrix A) + offset

#sum

lw $t9, ($t5) #load matrix A

lw $t8, ($t3) #load matrix B

lw $t6, ($t4) #load matrix C

add $t9, $t8, $t6 #add

sw $t9, ($t5)

addi $t0, $t0, 1

j for\_in\_add\_intr

for\_in\_end\_add\_intr:

addi $t1, $t1, 1

j for\_out\_add\_intr

for\_out\_end\_add\_intr:

jr $ra

######## MATRIX ADDITION LOOP UNROLLED########

## $a0 = base address of matrix B

## $a1 = base address of matrix C

## $a2 = base address of matrix A

## $a3 = N

matrix\_addition\_unr:

addi $t0, $0, 0 #i = 0

for\_out\_add\_unr:

beq $t0, $a3, for\_out\_end\_add\_unr #i! = N

addi $t1, $0, 0 #j = 0

for\_in\_add\_unr:

beq $t1, $a3, for\_in\_end\_add\_unr #j! = N

#calculate addresses

mul $t7, $t0, $a3 #i\*N

add $t7, $t7, $t1 #i\*N+j

sll $t7, $t7, 2 #(i\*N+j)\*4

add $t3, $t7, $a0 #(base address of matrix B) + offset

add $t4, $t7, $a1 #(base address of matrix C) + offset

add $t5, $t7, $a2 #(base address of resultant matrix A) + offset

#sum

lw $t9, ($t5) #load matrix A

lw $t8, ($t3) #load matrix B

lw $t6, ($t4) #load matrix C

add $t9, $t8, $t6 #add

sw $t9, ($t5)

addi $t1, $t1, 1

#calculate addresses

mul $t7, $t0, $a3 #i\*N

add $t7, $t7, $t1 #i\*N+j

sll $t7, $t7, 2 #(i\*N+j)\*4

add $t3, $t7, $a0 #(base address of matrix B) + offset

add $t4, $t7, $a1 #(base address of matrix C) + offset

add $t5, $t7, $a2 #(base address of resultant matrix A) + offset

#sum

lw $t9, ($t5) #load matrix A

lw $t8, ($t3) #load matrix B

lw $t6, ($t4) #load matrix C

add $t9, $t8, $t6 #add

sw $t9, ($t5)

addi $t1, $t1, 1

j for\_in\_add\_unr

for\_in\_end\_add\_unr:

addi $t0, $t0, 1

j for\_out\_add\_unr

for\_out\_end\_add\_unr:

jr $ra

#########################################

# #

# data segment #

# #

#########################################

.data

mat1: .asciiz "matrix1\_100.txt" # filename for matrix B

mat2: .asciiz "matrix2\_100.txt" # filename for matrix C

endl: .asciiz "\n"

spce: .asciiz " "

tabb: .asciiz "\t"

size: .word 100

stringBuff: .space 160000000