COMPSYS 304 Assignment 1

840454023, elee353

Q1.

(a)

# assumed signed

sra $14, $12, 7

(b)

#initialise 16 to be used

addi $8, $0, 16

#move to FP register

mtc1 $8, $f8

#convert to double

cvt.d.w $f8, $f8

#multiply and put to register

mul.d $f10, $f12, $f8

Q2.

# load index range

addi $t3, $0, 0

addi $t0, $0, 1000

# load base address of array A

add $t1, 0, $11

# load base address of array B

add $t2, 0, $12

L1:

# increment index by 4 each time

sll $t4, $t3, 2

# get index value of array A

add $t5, $t4, $t1

# get index value of array B

add $t6, $t4, $t2

# store the fourth byte of A in the first byte of B

lbu $t8, 3($t5) # load i-th element of array A into temporary register

sb $t8, 0($t6) # store the corresponding element of A into B

# store the third byte of A in the second byte of B

lbu $t8, 2($t5)

sb $t8, 1($6)

# store the second byte of A in the third byte of B

lbu $t8, 1($t5)

sb $t8, 2($t6)

# store the first byte of A to the fourth byte of B

lbu $t8, 0($t5)

sb $t8, 3($t6)

#increment index by 1

addi $t3, t3, 1

#decrement loop counter

addi $t0, $t0, -1

#check loop counter

bne $t0, $0, L1

jr $ra

Q3.

(a)

funct:

# store first argument to temporary register

add $t0, $a0, $0

#store 253 to temporary register, the max length of array

addi $t1, $a1, 253

#initialise the result register

addi $v0, $0, 0

# store ASCII ‘i’ to temporary register

addi $t5, $0, 0x151

# store ASCII ‘n’ to temporary register

addi $t6, $0, 0x156

# store ASCII ‘null’ to temporary register

addi $t7, $0, 0x00

L1:

# from temporary register

lbu $t2, 0($t0)

# test if it is ‘i’

bne $t2, $t5, L2

# load next

lbu $t3 ,1($t0)

#check t3 is null

beq $t7, $t3, L3

# test if the next character is ‘n’

bne $t3, $t6, L2

# increment result by 1.

addi $v0, $v0, 1

L2:

#increment to point to next char

addi $t0, $t0, 1

#decrement loop counter

addi $t1, $t1, -1

#check loop counter is not zero

bne $t1, $0, L1

#check t2 is not null

bne $t7, $t2, L1

L3:

jr $ra

(b)

.data

str1: .asciiz "in out in out "

.text

.globl main

main:

la $a0, str1 # array base address should be in $a0

ori $a1, $0, 50 # array size should be in $a1

jal funct

add $a0, $0, $v0

ori $v0, $0, 1

syscall # print the result

jr $ra

Q4.

(a)

funct:

# store the arguments into temporary register

# store x

add $t0, $a0, $0

#store y

add $t1, $a1, $0

# store n

add $t2, $a2, $0

#initialise the result register

addi $v0, $0, 0 # for z

addi $t7, $0, 0 # for 3x4

addi $t8, $0, 0 # for 2n

#store some integers to be used

addi $t3, $0, 1

addi $t4, $0, 3

addi $t5, $0, 4

addi $t6, $0, 2

addi $t7, $0, 10

addi $t8, $0, 7

# check x range

#check x < 10

slti $t9, $a0, $t7

bne $t9, $t3, Exit

# check !(x < 0)

slti $t9, $a0, $0

bne $t9, $0, Exit

# check y range

# check y < 7

slt $t10, $a1, $t8

bne $t10, $t3, Exit

# check !(y < 0)

slt $t10, $t8, $0

bne $t10, $0, Exit

#multiply x by 3 and put back to register

mul.s $t0, $t0, $t4

#multiply to get (3x)^2

mul.s $t0, $t0, $t0

#multiply to get (3x)^4

mul.s $t0, $t0, $t0

#shift left by two to get 2^n

sll $t6, $t6, $t2

#shift left by n for division

sll $t1, $t2

# z = 1

addi $vo, $vo, 1

#z = 1 + (3x)^4

add $vo, $vo, $t0

#z = 1 + (3x)^4+y/2^n

add $vo, $vo, $t1

Exit:

jr $ra

Q5.

calculate\_one\_element\_of\_Z:

# get the 5th argumentand put in in $t0. (size of row)

lw $t0, 0($sp)

#initialize FP register pairs $26, $27 with double value 1

addi $t6, $0, 1

mtc1 $t6, $f20

#convert to f26, f27 = 1.0

cvt.d.w $f26, f20

#initialize FP register pairs $28, $29 with double value 8

addi $t6, $0, 8

mtc1 $t6, $f20

#convert to f28, f29 = 8.0

cvt.d.w $f28, f20

# calculate the address of X[i][j]

# address of X[i][j] = base address of X + ((i \* size of row) + j) \* size of data

# (i \* size of row)

multu $a2, $t0

# no overflow so, HI = 0

mflo $t1

# ((i \* size of row) + j)

add $t1, $t1, $a3

# ((i \* size of row) + j) \* size of data

sll $t1, $t1, 3

addu $t1, $t1, $a0 # $t1 = address of X[i][j]

# read X[i][j] (the processor is little-endian)

# $f16,$f17 = X[i][j]

lwc1 $f16, 4($t1)

lwc1 $f17, 0($t1)

# $f16, $f17 = X[i][j] / 8

div.d $f16, $f16, $f28

# Y[i][j] = 1-(X[i][j]/8)

sub $26 ,$f26 , $16

swc1

# This function calculates Z[i][j]