COMPSYS 304 Assignment 1

840454023, elee353

Q1.

(a)

sra $14, $12, 7

(b)

addi $8, $0, 16 #initialise 16 to be used

mtc1 $8, $f8 #move to FP register

cvt.d.w $f8, $f8 #convert to double

mul.d $f10, $f12, $f8 #multiply and put to register

Q2.

addi $t3, $0, 0

addi $t0, $0, 1000

lw $t1, 0($11) # load base address of array A

lw $t2, 0($12) # load base address of array B

L1:

sll $t4, $t3, 2 # increment index by 4 each time

add $t1, $t4, $t1 # get index value of array A

add $t2, $t4, $t2 # get index value of array B

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

lbu $t5, 3($t1) # load i-th element of array A into temporary register

sb $t5, 0($t2) # store the corresponding element of A into B

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

lbu $t5, 2($t1)

sb $t5, 1($t2)

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

lbu $t5, 1($t1)

sb $t5, 2($t2)

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

lbu $t5, 0($t1)

sb $t5, 3($t2)

addi $t0, $t0, -1

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:

# load address of argument from temporary register

lbu $t2, 0($t0)

# test if it is ‘i’

bne $t2, $t5, L2

# increment address pointer by 1

addi $t3, $t2, 1

# test if the next character is ‘n’

bne $t3, $t6,

# 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

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 z

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

# z = z + 1

addi $t2, $t2, 1

# 3x

mtc1

//power\_of\_4:

Exit:

jr $ra

Q5.

***calculate\_one\_element\_of\_Z:***

**# This function calculates Z[i][j]**