# I don’t understand why my code stop after I enter first key on keyboard.

.data

asciiZero: .byte '0'

#exp: .byte 30

expBuffer: .space 60

expBufferlen: .word 0

.text

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

lui $t0, 0xffff # turn on TC

lw $t1, 8($t0)

ori $t1, $t1, 0x0001 # flip last bit

sw $t1, 8($t0) # put back value

lui $t0, 0xffff # trun on RC

lw $t1, 0($t0)

ori $t1, $t1, 0x0002

sw $t1, 0($t0)

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

la $s0, expBuffer # load array(expBuffer) to s0

lb $s1, 0($s0) # load address of s0 into s1

#

#loopnothing:

#addi $t3, $t3, 0

#j loopnothing

#

handle:

loop: # loop for polling values

jal retrieve

lw $a0, 0($v0) # set arg to a0 and call store function

jal store

beq $a0, 61, eva # if key input was "=" exit loop after store

j loop

eva: # call evaluate function

jal evaluate # convert value and find Z them put Z in expBuffer

jal print # display function

jr $ra

# polling read

retrieve:

#mfc0 $t0, $12

lui $t0, 0xffff

LOOP: # polling

lw $t1, 0($t0) # t1 sotre content RC

andi $t2, $t1, 0x0001

beq $t2, $zero, LOOP # check ready bit

lw $a0, 4($t0) # load the key pressed

jr $ra # return key

# store

store:

sb $a0, 0($s1) # save the input from key to buffer

addiu $s1, $s1, 1 # move address of array to next one

#expBufferlen++ ?

jr $ra

#...............end read

# X - "0" + Y = Z 0 = 48 9 = 57 '+ = 43' '= = 61'

evaluate:

lb $s2, 0($s0) # reload the address of array into s2

lb $a0, 0($s2)

jal char2num # convert X

sb $v0,0($t5) # sore vale of number X in to t5

add $s2, $s2, 2 # shift pointer of array to 'Y'

lb $a0, 0($s2) # call convert for Y

jal char2num # convert

sb $v0, 0($t6) # sore vale of number Y in to t6

add $t7, $t5, $t6 # Z at $t7

blt $t7, 10, go # if Z < 10, good to go

sub $t7, $t7, 10

li $t6, 1

lb $a0, 0($t6) # additional storing of '1' if Z >= 10

jal num2char

add $s1, $s1, 1 # move to next byte in array for storing

sb $v0, 0($s1) # save '1' into S1 array

go:

lb $a0, 0($t7) #

jal num2char # conver to char

lb $a0, 0($v0) # load number Z for store

jal store

#add $s1, $s1, 1 # move to next byte in array for storing

#sb $v0, 0($s1) # save Z into S1 array

jr $ra

# polling print

print:

lui $t0, 0xffff

lb $a0, 0($s1) # reload array again for print

LOOP1:

lw $t1, 8($t0)

andi $t2, $t1, 0x0001

beq $t2, $zero, LOOP1

sw $a0, 12($t0)

add $a0, $a0, 1 # go to next char in array

beq $a0, $zero, quit # check if end of array

j LOOP1

quit:

jr $ra # finsh print

# convert function

char2num:

lb $t0, asciiZero

subu $v0, $a0, $t0

jr $ra

num2char:

lb $t0, asciiZero

addu $v0, $a0, $t0

jr $ra

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

.ktext 0x80000080

sw $t0, \_k\_save\_t0 # store all registers

sw $t1, \_k\_save\_t1

sw $t2, \_k\_save\_t2

sw $t5, \_k\_save\_t5

sw $t6, \_k\_save\_t6

sw $t7, \_k\_save\_t7

sw $s0, \_k\_save\_s0

sw $s1, \_k\_save\_s1

sw $s2, \_k\_save\_s2

sw $a0, \_k\_save\_a0

sw $ra, \_k\_save\_ra

# check cause

mfc0 $k0, $13 # cause check

andi $k1, $k0, 0x007c # check if all digit is 1

bnez $k1, back # skip if not keyboard action

andi $k1, $k0, 0x0100 # check interrupt level

beqz $k1, back

# function call

la $k0, handle

jalr $k0

# clear causer reg

mtc0 $zero, $13 # clear Cause register

# reset status register

mfc0 $k0, $12 # status reg

andi $k0, 0x111D

ori $k0, 0x0001 # enable last bit of status

#ori $k0, 0x0002 # set 2nd last bit to 0

mtc0 $k0, $12 # put back

# return stage

back:

lw $t0, \_k\_save\_t0 # restore

lw $t1, \_k\_save\_t1

lw $t2, \_k\_save\_t2

lw $t5, \_k\_save\_t5

lw $t6, \_k\_save\_t6

lw $t7, \_k\_save\_t7

lw $s0, \_k\_save\_s0

lw $s1, \_k\_save\_s1

lw $s2, \_k\_save\_s2

lw $a0, \_k\_save\_a0

lw $ra, \_k\_save\_ra

eret #return to .text

.kdata

\_k\_save\_t0: .word 0

\_k\_save\_t1: .word 0

\_k\_save\_t2: .word 0

\_k\_save\_t5: .word 0

\_k\_save\_t6: .word 0

\_k\_save\_t7: .word 0

\_k\_save\_s0: .word 0

\_k\_save\_s1: .word 0

\_k\_save\_s2: .word 0

\_k\_save\_a0: .word 0

\_k\_save\_ra: .word 0