ECE3221 Lab 3

Stephen Cole

3553803

Source Code

.global \_start

\_start:

# MACROS

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

# PUSH

.macro push rx

addi sp,sp,-4

stw \rx,0(sp)

.endm

#POP

.macro pop rx

ldw \rx,0(sp)

addi sp,sp,4

.endm

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

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

Start:

movia sp,stacktop # initialize stack pointer

movia r23,0x10000000 # red led address

movia r22,0x10000010 # green led address

movia r21,0x10000020 # 7seg address

movia r20,0x10000050 # push button address

movia r19,lookup # 7segment lookup table

top:

call lock # close the lock

call reset # set progress counter to zero

call out7seg # display the count

# - - - - - - - - - - - - - - (a)

call getkey # r3 = button press = (0,1,2,3)

movia r7,code # r7 = addr of stored code digits

ldb r6,0(r7) # r6 = this code digit

bne r6,r3,fail # if( wrong key ) { fail }

call inc # count the successful keystroke

call out7seg # display the count

# - - - - - - - - - - - - - - (b)

movi r3,10 # reset r3

call delayN # wait for button

call getkey # set r3 = button

movia r7,code # set r7 = address of codes

ldb r6,1(r7) # load code

bne r6,r3,fail # check for correct key

call inc

call out7seg

# - - - - - - - - - - - - - - (c)

movi r3,10 # reset r3

call delayN # wait for button

call getkey # set r3 = button

movia r7,code # set r7 = address of codes

ldb r6,2(r7) # load code

bne r6,r3,fail # check for correct key

call inc

call out7seg

# - - - - - - - - - - - - - - (d)

movi r3,10 # reset r3

call delayN # wait for button

call getkey # set r3 = button

movia r7,code # set r7 = address of codes

ldb r6,3(r7) # load code

bne r6,r3,fail # check for correct key

call inc

call out7seg

call unlock # open the lock

call opentime # keep open for 10 seconds

br top # start over

fail:

ori r3,r0,7 # display F (fail)

call out7seg

ori r3,r0,50 # r3 = N wait N/10 sec

call delayN

br top

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

# Subroutines

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

# delay100ms RTervo May 2020

# returns after a 0.1 second delay

# using the DE2-115 interval timer

# input values: none

# no registers affected

delay100ms:

push r3

push r11

movia r11,0x10000000 # r11 = I/O base addr

# the interval counter is clocked at 50 MHz

# r3 = timing interval (clocks)

movia r3,5000000 # let delay = 0.1 sec

sthio r3,0x2008(r11) # set timer low half word

srli r3,r3,16 # see the other 16 bits

sthio r3,0x200C(r11) # set timer high half word

sthio r0,0x2000(r11) # TO=0 (clear timeout bit)

ori r3,r0,0x0004

sthio r3,0x2004(r11) # START=1, CONT=0, ITO=0

awaitTO:

ldhio r3,0x2000(r11)

andi r3,r3,1 # r3 = TO bit

beq r3,r0,awaitTO # wait for timeout

pop r11

pop r3

ret

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

# delayN SCole June 2020

# returns after a N/10 second delay

# input values: r3 == N

# no registers affected

delayN:

push ra # push registers into stack

push r3

push r6

push r7

mov r6, r0 # set inital values

movi r7, 0x64

delayloop:

call delay100ms

addi r3, r3, -1 # decrement r3

addi r6, r6, 1 # increment r6

beq r6, r7, enddelay # when r6 == max end

bgt r3, r0, delayloop # while N > 0 delay

enddelay:

pop r7 # pop registers from the stack

pop r6

pop r3

pop ra

ret

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

getkey:

push r6 # push needed registers into the stack

push r7

keytop:

ldwio r6,(r20) # load push buttons into r6

movi r3,0 # set r3 to the current value

movi r7,0x1 # set r7 to check for the first button

beq r6,r7,keyend # if the first button is pressed return r3

ldwio r6,(r20)

movi r3,1 # repeat for rest of the buttons

movi r7,0x2

beq r6,r7,keyend

ldwio r6,(r20)

movi r3,2

movi r7,0x4

beq r6,r7,keyend

ldwio r6,(r20)

movi r3,3

movi r7,0x8

beq r6,r7,keyend

br keytop # if no button is pressed repeat

keyend: # pop registers out of the stack

pop r7

pop r6

ret

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

#increment count

inc:

push r6 # push r6 into stack

movi r6,count # put count address into r6

ldw r3,(r6) # load count into r4

addi r3,r3,1 # inc

stw r3,(r6) # store new count

pop r6 # remove register from the stack

ret

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

reset:

push r6 # push r6 into stack

movia r6,count # put count address into r6

stw r0,(r6) # set count = 0

pop r6

ret

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

lock:

push r6 # push r6 into stack

movia r6,0xFFFFFFFF # set r6 high

stwio r6,(r23) # set red led high

stwio r0,(r22) # set green led high

pop r6

ret

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

unlock:

push r6 # push r6 into stack

movia r6,0xFFFFFFFF # set r6 high

stwio r0,(r23) # set red led low

stwio r6,(r22) # set green led high

pop r6

ret

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

out7seg:

push r6 # push onto stack

push r7

andi r6,r3,0x7 # force all but 3lsb low

add r7,r6,r19 # increment lookup address to corresponding bits

ldbu r6,(r7) # load values from lookup

stwio r6, (r21)

pop r6 # pop out of stack

pop r7

ret

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

opentime:

push r3 # push onto stack

push r6

push r7

push ra

movi r3,0x5 # set r3 = 5

movi r6,0 # set r6 = 0

movi r7,0xA # set r7 = A

opentop:

call out7seg # display 7seg

call delayN # delay based on r3

movi r3,0x6

call out7seg # display 7seg

movi r3,0x5

call delayN # delay based on r3

addi r6,r6,1

bne r6,r7,opentop

stwio r0,(r21) #display to 7seg

pop ra # pop registers

pop r7

pop r6

pop r3

ret

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

# ALLOCATION

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

# STACK

.skip 200

stacktop:

count: .word 0 # progress counter seen in hex display

code: .byte 3,2,0,1 # the four digits in the secret code

lookup: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x49, 0x5C, 0x63, 0x71 # lookup table for 7seg

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