#.include "ISR.s"

.global \_start

\_start:

.section .data

.equ Timer, 0xFF202000

.equ LED, 0xFF200000

.equ JP1 ,0xFF200060

.equ JTAG2, 0xFF201000

.equ LEDPeriod, 100000000 #3 second5F5E100

.section .text

movia r8,JP1

movia r9, 0xFFFFFFFF #turn everything off

stwio r9,0(r8)

movia r9, 0x07F557FF #set direction of or motors and

stwio r9,4(r8) #sensors to output and sensor data

#register to inputs

movia r16,JTAG2

movia r10,LED

stwio r0,0(r10)

Sensor0:

movia r10,0xFFFFFBFF #turns on sens0 valid

stwio r10,0(r8)

Poll0:

ldwio r11,0(r8) #load in r8 to r11

srli r11,r11,11 #mask to 11th bit sens0 valid in r12

andi r11,r11,1

bne r11,r0,Poll0

ldwio r11,0(r8)

srli r11,r11,27 #mask to result and store in r11

andi r11,r11,0xF

movi r12, 0x06

bgt r12,r11,InitializeInterrupts

br Poll0

InitializeInterrupts:

call IntInit #enable int

call JTAGInit #enable int in JTAG

call InvertLED

br WaitLoop

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

WaitLoop: br WaitLoop #waiting for interrupt from ps2 keyboard

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

call TimerInit

movi r15,0x31

movi r16,0x32

beq r14,r15,MOTOR0\_Forward

beq r14,r16,MOTOR1\_Forward

br WaitLoop

MOTOR0\_Forward:

movia r13,0xFFFFFFFE #E=1110

stwio r13,0(r8)

call CheckTimer

MOTOR0\_Reverse:

movia r13,0xFFFFFFFC #C=1100

stwio r13,0(r8)

call CheckTimer

br MOTOR0\_Forward

MOTOR1\_Forward:

movia r13,0xFFFFFFFB #B=1011

stwio r13,0(r8)

call CheckTimer

MOTOR1\_Reverse:

movia r13,0xFFFFFFF3 #3=0011

stwio r13,0(r8)

call CheckTimer

br MOTOR1\_Forward

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

#INITIALIZE

IntInit:

movui r10,0x100 #make bit 8 high for IRQ8

movui r11,0b1 #make bit 0 high for PIE bit

wrctl ienable,r10 #ienable/ctl3

wrctl ctl0,r11

ret

JTAGInit:

movi r14,0x1 #make bit 0 high to enable read interrupts

stwio r14,4(r16) #store at base+4

ret

TimerInit:

movia r17, Timer

movui r9, %lo(LEDPeriod)

stwio r9, 8(r17)

movui r9, %hi(LEDPeriod)

stwio r9, 12(r17)

stwio r0, 0(r17) #Clear Timeout

movui r9, 0x06

stwio r9, 4(r17) #Go, Not Continuous

ret

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

InvertLED:

movia r10,LED

ldw r11,0(r10)

andi r12,r11,0x01 #use LED0

beq r12,r0,ILOFF

andi r11,r11,0xFFFE #LED was on

br ILDONE

ILOFF: ori r11,r11,0x01

ILDONE: stw r11,0(r10)

ret

.section .exceptions, "ax"

.global ISR

ISR:

rdctl r11,ipending #read ipending

movi r13,0x100 #check if bit 8 is high to see if JTAG caused int

bne r11,r13,ExitISR #if not JTAG exit

POLL:

ldwio r14,0(r16) #if JTAG, poll for read value

andi r15,r14,0x8000 #mask bit 15 to see if read data is valid

beq r15,r0,POLL #if not keep polling

andi r14,r14,0xFF #if valid mask bits 0-7 for data itself

stwio r14,0(r16) #write data back into JTAG

movi r15,0x31

beq r14,r15,ContinueExit

movi r15,0x32

beq r14,r15,ContinueExit

ExitISR:

addi ea,ea,-4

eret

ContinueExit: eret

CheckTimer:

addi sp,sp,-4

stw ra,0(sp)

call TimerInit

ldw ra,0(sp)

addi sp,sp,4

TimerLoop:

ldw r9, 0(r17) #Read status of timer

andi r9, r9, 0x01 #Get only Timeout bit

beq r0, r9, TimerLoop #Not timed out yet

#Once timer times out

stwio r0, 0(r17) #Reset Timeout bit

ret