ECE243 Progress Report

Ultimate Boxer:
Force Sensitive Resistor avec VGA

Danja Papajani, KC Tremblay

8 April 2018

Changelog

Below is the changelog for our work on our ECE243 project.

Table 1.

Date	***Contributions***

March 30th, 2018	• KC
	  ○ Wrote first draft of VGA code
	  ○ Debugged with characters
	  ○ Filling screen halfway with characters

March 31st, 2018	• KC
	  ○ Able to fill whole screen with characters
	  ○ Debugged filling screen with pixels by drawing one colour to VGA
	• Danja
	  ○ Python script to convert JPG to BIN
	  ○ Wrote first draft of ADC/force sensitive resistor code

April 1st, 2018	• KC
	  ○ Debugged the way we accessed BIN file (drawing image sideways)
	  ○ VGA fully working without external cues
	• Danja
	  ○ Debugged ADC code
	  ○ Built hardware to connect to JP15
	  ○ Successfully reading values from the force sensor into specified registers

April 2nd, 2018 (Lab Session)	• KC
	  ○ Wrote working interrupt setup & ISR for PS2 keyboard
	  ○ Altered VGA code to be a subroutine

April 6th, 2018	• KC ○ Wrote keyboard input parser within ISR (displays “player 1”, “player 2”, or “player 3” based on whether key “1”, “2” or “3” key was pressed)

	• Danja
	  ○ Wrote interrupt to output audio files
	  ○ Added thresholds for ADC input

April 7th, 2018	• KC
	  ○ Combined VGA & PS2 Keyboard to work together
	• Danja
	  ○ Further debugged interrupt for audio core
	  ○ Draft of combined code with ADC and audio interrupt
	  ○ Altered ADC code to include subroutine

April 8th, 2018	• KC
	  ○ Wrote subroutine to display meter level indicating how hard user pressed force sensitive resistor
	• Danja
	  ○ Added ability to keep and maintain ongoing score through counters
	  ○ Sends information into VGA controller
	• KC & Danja
	  ○ Combined our respective code

Appendix A

Hardware Setup

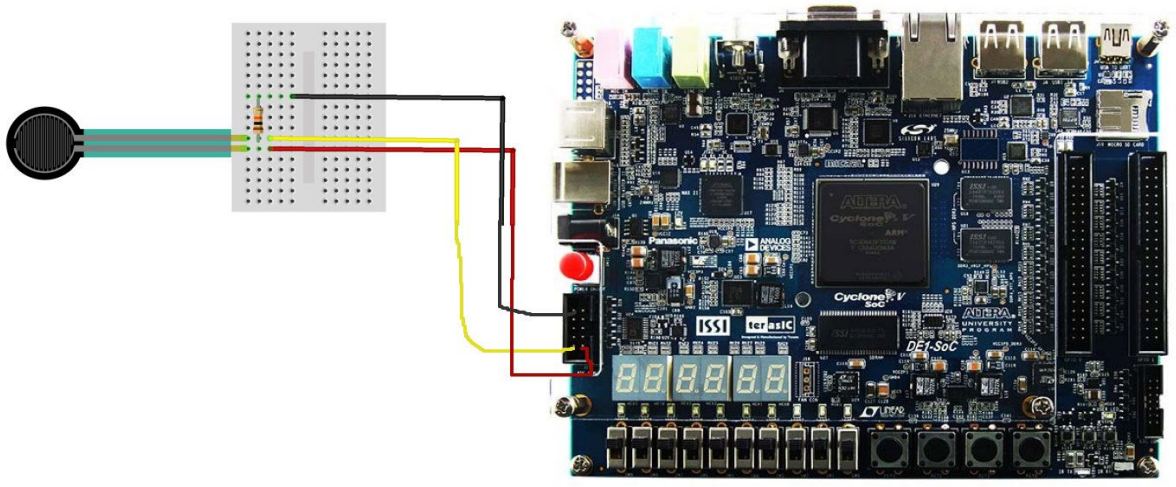

Figure 1.

Appendix B

Python script, courtesy of
https://gist.github.com/gnrr/7065820/revisions#diff-c289da366388679a8e176cd3969df550

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

from PIL import Image
import struct, os, sys

def usage():
    print './png2rgb565.py HOGE.png'
    sys.exit(1)

def error(msg):
    print msg
    sys.exit(-1)

def write_bin(f, pixel_list):
    for pix in pixel_list:
        r = (pix[0] >> 3) & 0x1F
        g = (pix[1] >> 2) & 0x3F
        b = (pix[2] >> 3) & 0x1F
        f.write(struct.pack('H', (r << 11) + (g << 5) + b))

##
if __name__ == '__main__':
    args = sys.argv
    if len(args) != 2: usage()
    in_path = args[1]
    if os.path.exists(in_path) == False: error('not exists: ' + in_path)

    body, _ = os.path.splitext(in_path)
    out_path = body + '.bin'

    img = Image.open(in_path).convert('RGB')
    img.save('new.jpg')
    pixels = list(img.getdata())
    # print pixels

    with open(out_path, 'wb') as f:
        write_bin(f, pixels)
```

Appendix C

Audio Code

```
.section .text

.equ AUDIO, 0xFF203040

#include wav file
SONG:
.incbin "priceIsRight.wav"

#used to store 32bits/sample
SAMPLE_SOUND:
    .align 2
    .skip 266276

.global _start

_start:

    #move location of audio core into r11
    movia r11, AUDIO

    #move location of song file into r8
    movia r9, SONG

    #move location of sampled sound into r10
    movia r10, SAMPLE_SOUND
    addi r10, r10, 4

    #load hald word sample of the song file
    ldh r12, 34(r9)
    #get data size in bytes
    ldw r13, 40(r9)
    #get first 1 bits of data
    ldb r14, 44(r9)

    #mov r9 to data
    addi r9, r9, 44


ACTIVATE_INTERRUPTS:
```

```
    #activate audio codec interrupts
    #write binary 10 into audio core for read interrupt
    #unsigned number allows interrupt enable for left and right FIFO
    movui r3, 0x2
    stwio r3, 0(r11)

    #make all bits 0 except 6th bit
    #enabling ctl3
    movui r3, 0x40
    wrctl ctl3, r3

############################################################
############                          #################
############      SOUND INTERRUPT     #################
############                          #################
############################################################
.section .exceptions, "ax"

ISR:

    #check if core caused an interrupt
    #check the 6th bit
    rdctl et, ctl4
    andi et, et, 0x40
    bne et, r0, WRITE_TO_CORE

br EXIT

WRITE_TO_CORE:

    #write to left and right channels
    ldw r19, 0(r9)
    stwio r19, 8(r11)
    stwio r19, 12(r11)
    addi r9, r9, 4

br EXIT

EXIT:
    subi ea, ea, 4
    eret
```

Appendix D

Main Code

```
.text

      .equ ADDR_VGA, 0x08000000
      .equ ADDR_CHAR, 0x09000000
      .equ TOP_OF_MEMORY, 0x1000
      .equ ADDR_KEYBOARD, 0xFF200100
    .equ ADC, 0xFF204000


.data

BACKGROUND:
      .incbin "BEEP.bin"

.global _start

_start:

    #total score
    movi r17, 0x0
    #initalize counters to 0
    movi r18, 0x0
    movi r19, 0x0
    movi r20, 0x0


#########################################################
############                          ##################
############            MAIN          ##################
############                          ##################
#########################################################
MAIN:

    #initialize  registers with respective I/O device
      movia r2, ADDR_VGA
      movia r3, ADDR_CHAR
      movia r15, BACKGROUND
      movia r16, ADC
      movia r8, ADDR_KEYBOARD
      movia sp, TOP_OF_MEMORY
```

```
    #overwrite character buffer for player
    movi r6, 0x20
    stbio r6, 3346(r3)
    movi r6, 0x20
    stbio r6, 3347(r3)
    movi r6, 0x20
    stbio r6, 3348(r3)
    movi r6, 0x20
    stbio r6, 3349(r3)
    movi r6, 0x20
    stbio r6, 3350(r3)
    movi r6, 0x20
    stbio r6, 3351(r3)
    movi r6, 0x20
    stbio r6, 3352(r3)
    movi r6, 0x20
    stbio r6, 3353(r3)

    #overwrite character buffer for score
    movi r6, 0x20
    stbio r6, 3378(r3)
    movi r6, 0x20
    stbio r6, 3379(r3)
    movi r6, 0x20
    stbio r6, 3380(r3)

WRITE:

    call VGA_SUBROUTINE

    #trigger update for analog to digital converter
    movi r21, 0x01
    stwio r21, 0(r16)

    #enable interrupt on PS2 Keyboard
    movui r9, 0b1         # bit 0 of controller is "read interrupt enable"
    stwio r9, 4(r8)

    # enable external interrupts
    movi r9, 0b10000000
    wrctl ienable, r9     # ienable (ctl3)
```

```
    # enable external interrupts
    movi r9, 0b1
    wrctl status, r9                  # status (ctl0) , PIE = 1

    call READ

write_:

    #move analog information into different register
    mov r22, r21

    #branch to respective pressure level
    movi r21, 0x100
    beq r22, r21, LIGHT

    movi r21, 0x824
    beq r22, r21, MEDIUM

    movi r21, 0xE23
    beq r22, r21, HEAVY

READ:

    #recieve analog information from for sensor
    ldwio r21, 0(r16)

    #write 0 to update on ADC
    movi r23, 0x0
    stwio r23, 0(r16)
    ret

LIGHT:

    #increment counter for light pressure
    addi r18, r18, 1
    movi r21, 0x01
    beq r18, r21, UPDATE_SCORE_LIGHT

br MAIN

UPDATE_SCORE_LIGHT:
```

```
    #update final score
    addi r17, r17,3
    #reset counter
    movi r18, 0x0

br MAIN

MEDIUM:

    #increment counter for medium pressure
    addi r19, r19, 1
    movi r21, 0x03
    beq r19, r21, UPDATE_SCORE_MEDIUM

br MAIN
```

```
#########################################################
############                           #################
############    METER_SUBROUTINE       #################
############                           #################
#########################################################
METER_SUBROUTINE:
      ####### r6 -> maximum number of columns
      ####### r7 -> maximum number of rows
      ####### r8 -> iterator for inner loop (j)
      ####### r9 -> pixel colour to be written to vga
      ####### r10 -> iterator for outer loop (i)
      ####### r11 -> vga address offset
      ####### r12 -> result of math for i value (x position)
      ####### r13 -> result of math for j value (y position)


      #PROLOGUE
      movia sp, TOP_OF_MEMORY
      subi sp, sp, 36
      stw r5, 0(sp)
      stw r6, 4(sp)
      stw r7, 8(sp)
      stw r8, 12(sp)
      stw r9, 16(sp)
      stw r10, 20(sp)
```

```
      stw r11, 24(sp)
      stw r12, 28(sp)
      stw r13, 32(sp)

      #######DOUBLE FOR LOOP FOR DISPLAYING IMAGE######

      movui r9, 0x0FE0
      movi r6, 174
      movi r10, 129
      movi r8, 18

OUTERFORLOOP_1:

    bge r10, r6, BYE_1          # stop plotting when i = max y value
(passed in)
    movi r8, 18                 # restart inner for loop iterator (j)

INNERFORLOOP_1:

    movia r2, ADDR_VGA
    bge r8, r7, EXITINNER_1     # go to outer for loop when the row has
been drawn
    muli r12, r10, 1024         # multiply y coordinate value by 1024
(given equation)
    muli r13, r8, 2             # multiply x coordinate value by 2
    add r11, r12, r13           # given equation to find address offset
    add r2, r2, r11             # add offset to vga address
    sthio r9, 0(r2)             # write pixel to vga
    addi r8, r8, 1

br INNERFORLOOP_1


EXITINNER_1:

    addi r10, r10, 1            # increment outer loop iterator

br OUTERFORLOOP_1


################################################################################
#
```

```
      #EPILOGUE
BYE_1:

      ldw r13, 32(sp)
      ldw r12, 28(sp)
      ldw r11, 24(sp)
      ldw r10, 20(sp)
      ldw r9, 16(sp)
      ldw r8, 12(sp)
      ldw r7, 8(sp)
      ldw r6, 4(sp)
      ldw r5, 0(sp)
      addi sp, sp, 36
      ret

#########################################################
############                          ##################
############     VGA_SUBROUTINE       ##################
############                          ##################
#########################################################
VGA_SUBROUTINE:

      ####### r6 -> maximum number of columns
      ####### r7 -> maximum number of rows
      ####### r8 -> iterator for inner loop (j)
      ####### r9 -> pixel colour to be written to vga
      ####### r10 -> iterator for outer loop (i)
      ####### r11 -> vga address offset
      ####### r12 -> result of math for i value (x position)
      ####### r13 -> result of math for j value (y position)
      ####### r15 -> holds address of .bin image



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

      #PROLOGUE
      movia sp, TOP_OF_MEMORY
      subi sp, sp, 40
      stw r15, 0(sp)
      stw r5, 4(sp)
```

```
      stw r6, 8(sp)
      stw r7, 12(sp)
      stw r8, 16(sp)
      stw r9, 20(sp)
      stw r10, 24(sp)
      stw r11, 28(sp)
      stw r12, 32(sp)
      stw r13, 36(sp)

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


      #######DOUBLE FOR LOOP FOR DISPLAYING IMAGE######


      movi r6, 240                        # max y value
      movi r7, 320                        # max x value
      mov r10, r0


OUTERFORLOOP:

    bge r10, r6, BYE          # stop plotting when i = 239
    mov r8, r0                  # restart inner for loop iterator (j)

INNERFORLOOP:

    movia r2, ADDR_VGA
    bge r8, r7, EXITINNER       # go to outer for loop when the row has
been drawn
    muli r12, r10, 1024         # multiply y coordinate value by 1024
(given equation)
    muli r13, r8, 2             # multiply x coordinate value by 2
    add r11, r12, r13           # given equation to find address offset

    ldh r9, 0(r15)              # get content to write to vga
    addi r15, r15, 2            # increment by 2 bytes to get next pixel to
display

    add r2, r2, r11             # add offset to vga address
    sthio r9, 0(r2)             # write pixel to vga
    addi r8, r8, 1
```

```
br INNERFORLOOP


EXITINNER:

    addi r10, r10, 1            # increment outer loop iterator

br OUTERFORLOOP


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

      #EPILOGUE
      BYE:
      ldw r13, 36(sp)
      ldw r12, 32(sp)
      ldw r11, 28(sp)
      ldw r10, 24(sp)
      ldw r9, 20(sp)
      ldw r8, 16(sp)
      ldw r7, 12(sp)
      ldw r6, 8(sp)
      ldw r5, 4(sp)
      ldw r15, 0(sp)
      addi sp, sp, 40
      ret

#########################################################
###########                           ################
########### INTERRUPT SERVICE ROUTINE ################
###########          (ISR)            ################
#########################################################
.section   .exceptions,  "ax"     # ensures ISR is at 0x20

KEYBOARD_ISR:

    # PROLOGUE
    movia sp, TOP_OF_MEMORY
    subi sp, sp, 40         # save 8 registers on the stack
    stw r14, 0(sp)          # save r2 (address of character buffer)
    stw r3, 4(sp)           # save r3 (address of keyboard)
```

```
    stw r4, 8(sp)           # save r4 (data read from keyboard)
    stw r5, 12(sp)          # save r5 (holds true/false if data = key
value)
    stw r6, 16(sp)          # save r6 (value to be displayed to vga)
    stw r17, 20(sp)
    stw r18, 24(sp)
    stw r19, 28(sp)
    stw r20, 32(sp)

    rdctl et, ipending        # check ipending (ctl4)
    andi et, et, 0x80         # if (IRQ7 ! = 1), no interrupt then exit
    beq et, r0, EXIT


    ######## CODE TO DISPLAY CHARACTER TO VGA HERE ##########

    movia r14, ADDR_CHAR
    movia r3, ADDR_KEYBOARD
    ldbio r4, 0(r3)           # load the data (character) being read from
the keyboad
    andi r4, r4, 0xFF         # store only the data (first 7-0 bits)

    ########### FIGURE OUT WHICH KEY WAS PRESSED ############

CHECK_KEY_1:

    cmpeqi r5, r4, 0x16
    beq r5, r0, CHECK_KEY_2

    movi r6, 0x50                   # P in hex
    stbio r6, 3346(r14)             # load to vga
    movi r6, 0x4C                   # L
    stbio r6, 3347(r14)
    movi r6, 0x41                   # A
    stbio r6, 3348(r14)
    movi r6, 0x79                   # Y
    stbio r6, 3349(r14)
    movi r6, 0x45                   # E
    stbio r6, 3350(r14)
    movi r6, 0x52                   # R
    stbio r6, 3351(r14)
    movi r6, 0x20                   # ' '
```

```
    stbio r6, 3352(r14)
    movi r6, 0x31                   # 1
    stbio r6, 3353(r14)

CHECK_KEY_2:

    cmpeqi r5, r4, 0x1E
    beq r5, r0, CHECK_KEY_3

    movi r6, 0x50          # P in hex
    stbio r6, 3346(r14)    # load to vga
    movi r6, 0x4C          # L
    stbio r6, 3347(r14)
    movi r6, 0x41          # A
    stbio r6, 3348(r14)
    movi r6, 0x79          # Y
    stbio r6, 3349(r14)
    movi r6, 0x45          # E
    stbio r6, 3350(r14)
    movi r6, 0x52          # R
    stbio r6, 3351(r14)
    movi r6, 0x20          # ' '
    stbio r6, 3352(r14)
    movi r6, 0x32          # 2
    stbio r6, 3353(r14)

CHECK_KEY_3:

    cmpeqi r5, r4, 0x26
    beq r5, r0, CHECK_SPACE_KEY

    movi r6, 0x50          # P in hex
    stbio r6, 3346(r14)    # load to vga
    movi r6, 0x4C          # L
    stbio r6, 3347(r14)
    movi r6, 0x41          # A
    stbio r6, 3348(r14)
    movi r6, 0x79          # Y
    stbio r6, 3349(r14)
    movi r6, 0x45          # E
    stbio r6, 3350(r14)
    movi r6, 0x52          # R
```

```
    stbio r6, 3351(r14)
    movi r6, 0x20          # ' '
    stbio r6, 3352(r14)
    movi r6, 0x33          # 3
    stbio r6, 3353(r14)

CHECK_SPACE_KEY:

    cmpeqi r5, r4, 0x29
    beq r5, r0, EXIT

    movi r6, 10
    bgt r17, r6, CHECKSCORE
    movi r6, 4
    bgt r17, r6, LIGHTSCORE
    blt r17, r6, EXIT

LIGHTSCORE:

   movi r6, 0x31
   stbio r6, 3378(r14)
   movi r6, 0x30
   stbio r6, 3379(r14)
   movi r6, 0x30
   stbio r6, 3380(r14)

   movi r7, 125
   call METER_SUBROUTINE

br EXIT

CHECKSCORE:

    movi r6, 20
    blt r17, r6, UPDATELIGHT
    bgt r17, r6, UPDATEHIGH

UPDATELIGHT:

    movi r6, 0x33
    stbio r6, 3378(r14)
    movi r6, 0x32
```

```
    stbio r6, 3379(r14)
    movi r6, 0x35
    stbio r6, 3380(r14)

    movi r7, 220
    call METER_SUBROUTINE

br EXIT

UPDATEHIGH:

    movi r6, 0x35
    stbio r6, 3378(r14)
    movi r6, 0x30
    stbio r6, 3379(r14)
    movi r6, 0x30
    stbio r6, 3380(r14)

    movi r7, 300
    call METER_SUBROUTINE

br EXIT
#############################################################################
EXIT:

    #total score
    movi r17, 0x0
    #initalize counters to 0
    movi r18, 0x0
    movi r19, 0x0
    movi r20, 0x0

    #EPILOGUE
    ldw r20, 32(sp)
    ldw r19, 28(sp)
    ldw r18, 24(sp)
    ldw r17, 20(sp)
    ldw r6, 16(sp)
    ldw r5, 12(sp)
    ldw r4, 8(sp)
    ldw r3, 4(sp)
    ldw r14, 0(sp)
```

```
    addi sp, sp, 40

    subi ea, ea, 4 # make sure we execute the instruction that was
interrupted
    eret           # exception return
```