.text

## Register assignments

# $r0 - 0

# $r1 - volatile

# $r2 - volatile

# $r3 - volatile

# $r4 - volatile

# $r5 - volatile

# $r6 - volatile

# $r7 - volatile

# $r8 - computation parameter 1

# $r9 - computation parameter 2

# $r10 - computation result

# $r22 - previous keyboard input

# $r23 - program flow stack pointer

# $r24 - old pointer location

# $r25 - current pointer location

# $r26 - pen down (indicates whether to draw a pixel or not

# $r27 - stack pointer

# $r28 - frame pointer

# $r29 - drawing color

# $r30 - keyboard input

# $r31 - return address

# The stack pointer will be used solely for remembering previous pixel state. This may change.

# Pixels are stored in the following format [25:8] = Pixel index (Address), [7:0] = color index

# The custi1 and custi2 operations are used for storing to the display and loading from the display memory, respectively

## Begin initialization routine

init:

lw $r27, pixelMemBegin($r0) # initialize the stack pointer

nop

nop

nop

nop

lw $r28, pixelMemBegin($r0) # initialize the frame pointer

nop

nop

nop

nop

lw $r23, programMemBegin($r0) # initialize the program flow stack pointer

nop

nop

nop

nop

addi $r25, $r0, 8000 # set the initial pointer position (this position is currently arbitrary)

nop

nop

nop

nop

jal populateTopMenu

nop

nop

nop

nop

jal drawSelectedLine

nop

nop

nop

nop

j test # change this to choose which loop to jump to

nop

nop

nop

nop

## End initialization routine

nop

nop

nop

nop

nop

nop

nop

nop

## Begin test program flow. Uncomment first instruction to run main below

nop

nop

nop

nop

nop

nop

nop

nop

test:

nop

nop

nop

nop

jal checkKeys

nop

nop

nop

nop

jal updateCursor

nop

nop

nop

nop

#jal drawSelectedLine

nop

nop

nop

nop

j test

nop

nop

nop

nop

nop

nop

nop

nop

## End test program flow

nop

nop

nop

nop

nop

nop

nop

nop

## Begin main program flow

nop

nop

nop

nop

nop

nop

nop

nop

main: #The main program flow

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

j main

nop

nop

nop

nop

nop

nop

nop

nop

## End main program flow

nop

nop

nop

nop

nop

nop

nop

nop

## Begin color toggling code

nop

nop

nop

nop

nop

nop

nop

nop

incrementColor: # Increment the index of the color to be drawn. loop at 0

nop

nop

nop

nop

lw $r1, numColors($r0) # Load the number of colors

nop

nop

nop

nop

blt $r29, $r1, 2 # If the drawing color is less than the number of colors, increment the drawing color

nop

nop

nop

nop

addi $r29, $r0, 0# Otherwise set the drawing color to 0

nop

nop

nop

nop

ret

nop

nop

nop

nop

addi $r29, $r29, 1

nop

nop

nop

nop

ret

nop

nop

nop

nop

nop

nop

nop

nop

decrementColor: #Decrement the index of the color to be drawn. loop at 0

nop

nop

nop

nop

lw $r1, numColors($r0) # Load the number of colors

nop

nop

nop

nop

blt $r29, $r0, 0x2 # If the color index is less than 0, set it to max # colors -1

nop

nop

nop

nop

addi $r29, $r29, -1 # Otherwise decrement the drawing color

nop

nop

nop

nop

ret

nop

nop

nop

nop

addi $r1,$r1,-1

nop

nop

nop

nop

addi $r29, $r1, 0

nop

nop

nop

nop

ret

nop

nop

nop

nop

nop

nop

nop

nop

## End color toggling code

nop

nop

nop

nop

nop

nop

nop

nop

## Begin cursor drawing code

nop

nop

nop

nop

nop

nop

nop

nop

updateCursor: # Updates the cursor location. If the cursor has moved, fills in old space

nop

nop

nop

nop

bne $r25, $r24, drawCursor # If the current position doesn't equal the old position, draw the cursor

nop

nop

nop

nop

ret

nop

nop

nop

nop

nop

nop

nop

nop

drawCursor:

nop

nop

nop

nop

#First fill in old cursor area with data from memory

nop

nop

nop

nop

#While we haven't emptied the cursor stack, pop off a pixel and write it to the display

nop

nop

nop

nop

fillOld: blt $r27, $r28, doneFilling # if our current stack pointer is less than the frame pointer, we're done

nop

nop

nop

nop

lw $r2, 0($r27) # Load the pixel at our current stack pointer

nop

nop

nop

nop

sra $r3, $r2, 8 # Extract the pixel address

nop

nop

nop

nop

addi $r5, $r0, 255 #Create a mask for extracting color

nop

nop

nop

nop

or $r4, $r2, $r5 # Extract the color index of the pixel

nop

nop

nop

nop

custi1 $r3, $r2, 0 #Write this old color data to the display

nop

nop

nop

nop

addi $r27,$r27,-1 # Subtract 1 from the stack pointer

nop

nop

nop

nop

j fillOld

nop

nop

nop

nop

nop

nop

nop

nop

doneFilling: # If we're done filling in the old cursor, fill in the new cursor

nop

nop

nop

nop

addi $r27, $r28, 0 # set our stack pointer to the frame pointer

nop

nop

nop

nop

lw $r2, cursorColor($r0) # load the cursor color

nop

nop

nop

nop

custi2 $r3, $r25,0 # load the old pixel color

nop

nop

nop

nop

sll $r4, $r25, 8 # shift over the pixel address

nop

nop

nop

nop

or $r4, $r4, $r3 # or the bits together to create the pixel representation

nop

nop

nop

nop

sw $r4, 0($r27) # store the pixel onto the stack

nop

nop

nop

nop

addi $r27, $r27, 1 # increment the stack pointer

nop

nop

nop

nop

nop

nop

nop

nop

#Then draw the new cursor

nop

nop

nop

nop

custi1 $r2, $r25, 0 # fill the location of the current cursor with a color

nop

nop

nop

nop

nop

nop

nop

nop

#If the pen is down, fill the space where the pen used to be with the drawing color

nop

nop

nop

nop

addi $r5, $r0, 1

nop

nop

nop

nop

blt $r26, $r5, updateOldCursor #If the pen down value is 0 or less, don't store the pixel value

nop

nop

nop

nop

custi1 $r24, $r29, 0 #Otherwise, set the previous pixel's value to be the color value that we are writing

nop

nop

nop

nop

#TODO for bigger line widths, just modify the fill code to fill the color instead of the old color and take off the above line

nop

nop

nop

nop

#Set the old cursor value to be the current cursor value

nop

nop

nop

nop

updateOldCursor: addi $r24, $r25,0

nop

nop

nop

nop

ret

nop

nop

nop

nop

nop

nop

nop

nop

## End cursor drawing code

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

## Start keyboard button checking

nop

nop

nop

nop

nop

nop

nop

nop

checkKeys:

nop

nop

nop

nop

bne $r22, $r30, continueChecking # check for change in input

nop

nop

nop

nop

ret # if not, do nothing

nop

nop

nop

nop

continueChecking:

nop

nop

nop

nop

lw $r2, maxPixelIndex($r0) # $r2 = 307200

nop

nop

nop

nop

lw $r3, numReservedPixels($r0) # $r3 = 25600

nop

nop

nop

nop

sub $r4, $r2, $r3 # $r4 = 307200 - 25600 = max number of usable pixels

nop

nop

nop

nop

checkUp:

nop

nop

nop

nop

addi $r1, $r0, 42

nop

nop

nop

nop

bne $r30, $r1, checkDown

nop

nop

nop

nop

addi $r25, $r25, -640 #up

nop

nop

nop

nop

j checkedInput

nop

nop

nop

nop

checkDown:

nop

nop

nop

nop

addi $r1, $r0, 36

nop

nop

nop

nop

bne $r30, $r1, checkLeft

nop

nop

nop

nop

addi $r25, $r25, 640 #down

nop

nop

nop

nop

j checkedInput

nop

nop

nop

nop

checkLeft:

nop

nop

nop

nop

addi $r1, $r0, 22

nop

nop

nop

nop

bne $r30, $r1, checkRight

nop

nop

nop

nop

addi $r25, $r25, -1 #left

nop

nop

nop

nop

j checkedInput

nop

nop

nop

nop

nop

nop

nop

nop

checkRight:

nop

nop

nop

nop

addi $r1, $r0, 40

nop

nop

nop

nop

bne $r30, $r1, checkInsert

nop

nop

nop

nop

addi $r25, $r25, 1 #right

nop

nop

nop

nop

j checkedInput

nop

nop

nop

nop

nop

nop

nop

nop

checkInsert:

nop

nop

nop

nop

addi $r1, $r0, 32

nop

nop

nop

nop

bne $r30, $r1, checkHome

nop

nop

nop

nop

#If we pressed insert, toggle whether the pen is down or not

nop

nop

nop

nop

bne $r26, $r0, setPenUp

nop

nop

nop

nop

addi $r26, $r0, 1

nop

nop

nop

nop

j checkedInput

nop

nop

nop

nop

setPenUp: addi $r26, $r0,0

nop

nop

nop

nop

j checkedInput

nop

nop

nop

nop

nop

nop

nop

nop

checkHome:

nop

nop

nop

nop

addi $r1, $r0, 24

nop

nop

nop

nop

bne $r30, $r1, checkedInput

nop

nop

nop

nop

#If we pressed home, increment the color that we are drawing with the pen

nop

nop

nop

nop

addi $r27, $r27, 1 # Increment the stack pointer

nop

nop

nop

nop

sw $r31, 0($r27) #store the return address

nop

nop

nop

nop

jal incrementColor

nop

nop

nop

nop

lw $r31, 0($r27) #load the return address

nop

nop

nop

nop

addi $r27, $r27, -1 # Decrement the stack pointer

nop

nop

nop

nop

j checkedInput

nop

nop

nop

nop

nop

nop

nop

nop

checkedInput:

nop

nop

nop

nop

add $r22, $r30, $r0 # set last pressed key

nop

nop

nop

nop

blt $r25, $r3, wrapBegin2End # if $r25<25600, add number of usable pixels. Too high up

nop

nop

nop

nop

blt $r2, $r25, wrapEnd2Begin # if $r25>307200, subtract number of usable pixels. Too low down

nop

nop

nop

nop

ret # else return

nop

nop

nop

nop

wrapBegin2End: add $r25, $r25, $r4

nop

nop

nop

nop

ret

nop

nop

nop

nop

wrapEnd2Begin: sub $r25, $r25, $r4

nop

nop

nop

nop

ret

nop

nop

nop

nop

nop

nop

nop

nop

## End keyboard button checking

nop

nop

nop

nop

nop

nop

nop

nop

## Begin top menu population

nop

nop

nop

nop

nop

nop

nop

nop

populateTopMenu:

nop

nop

nop

nop

# For each available color

nop

nop

nop

nop

lw $r2, numColors($r0) # Load the number of colors

nop

nop

nop

nop

lw $r5, topFeatureDimension($r0) # Load the dimension of the top feature

nop

nop

nop

nop

addi $r1, $r0, 0 # Counter for the current drawing color index

nop

nop

nop

nop

addi $r3, $r0, 0 # Counter for the current pixel in the row

nop

nop

nop

nop

addi $r4, $r0, 0 # Counter for the count within a square

nop

nop

nop

nop

addi $r6, $r0, 2560 # Counter for the current row pixel start position. Start on row 4

nop

nop

nop

nop

add $r8, $r5, $r5 # make 2\* the top feature dimension so we know when to change the color

nop

nop

nop

nop

drawLine: blt $r2,$r1, finishLineDraw # If we've passed the max # of colors, the line is finished

nop

nop

nop

nop

addi $r3, $r0, 0 # Zero the counter for the inter-row count

nop

nop

nop

nop

startColorLine: addi $r4, $r0, 0 # Zero the counter for within square count

nop

nop

nop

nop

colorLine:

nop

nop

nop

nop

add $r7, $r6, $r3 # Add the row pixel position and the pixel start position to get the current pixel position

nop

nop

nop

nop

#If our position is less than the feature position, draw black (0)

nop

nop

nop

nop

blt $r4, $r5, blackLineDraw

nop

nop

nop

nop

# If we're less than 2x, draw the current color

nop

nop

nop

nop

blt $r4, $r8, colorLineDraw

nop

nop

nop

nop

#Otherwise, increment color index, reset counts

nop

nop

nop

nop

addi $r1, $r1, 1

nop

nop

nop

nop

addi $r4, $r0, 0

nop

nop

nop

nop

j drawLine

nop

nop

nop

nop

blackLineDraw:

nop

nop

nop

nop

custi1 $r1, $r0, 0 # Store black

nop

nop

nop

nop

j drawFinished

nop

nop

nop

nop

colorLineDraw:

nop

nop

nop

nop

custi1 $r1, $r6, 0 # Store the current color in the current pixel location

nop

nop

nop

nop

j drawFinished

nop

nop

nop

nop

nop

nop

nop

nop

drawFinished:

nop

nop

nop

nop

addi $r3, $r3, 1 #increment our count within the row

nop

nop

nop

nop

addi $r4, $r4, 1 #increment our count within our color

nop

nop

nop

nop

j colorLine

nop

nop

nop

nop

# Draw alternating empty pixels for the feature size followed by color pixels for the feature size

nop

nop

nop

nop

finishLineDraw:

nop

nop

nop

nop

addi $r6, $r6, 640 # Move to the next row

nop

nop

nop

nop

addi $r1, $r0, 0# Reset our color count

nop

nop

nop

nop

addi $r9, $r0, 640

nop

nop

nop

nop

mul $r9, $r5, $r9

nop

nop

nop

nop

blt $r6, $r9, drawLine # If we are less than our max pixel count, keep drawing the line

nop

nop

nop

nop

ret

nop

nop

nop

nop

nop

nop

nop

nop

## End top menu population

nop

nop

nop

nop

nop

nop

nop

nop

## Draw a selected line depending on what color is being drawn

nop

nop

nop

nop

nop

nop

nop

nop

drawSelectedLine:

nop

nop

nop

nop

#First clear all the pixels in the selected row. Then draw the line underneath the selected color

nop

nop

nop

nop

addi $r1, $r0, 15360 # the starting pixel of the selected row

nop

nop

nop

nop

addi $r2, $r0, 640 # the number of pixels in a row

nop

nop

nop

nop

addi $r3, $r0, 0 # Zero the pixel we're currently on

nop

nop

nop

nop

blackRow: blt $r3, $r2, writeBlack #Write the black row

nop

nop

nop

nop

ret

nop

nop

nop

nop

j selectedLineDrawing

nop

nop

nop

nop

writeBlack:

nop

nop

nop

nop

add $r4, $r3, $r1 # Find our current location

nop

nop

nop

nop

custi1 $r0, $r4, 0 # Write black into the location

nop

nop

nop

nop

addi $r3, $r3, 1 # Increment our position in our current row

nop

nop

nop

nop

j blackRow

nop

nop

nop

nop

selectedLineDrawing: # Draw the selected line

nop

nop

nop

nop

# First figure out our starting position. Then draw the color for 30 pixels

nop

nop

nop

nop

lw $r5, topFeatureDimension($r0) # Load the top feature dimension

nop

nop

nop

nop

addi $r6, $r0, 60 # Get the dimension. Possible TODO make this better

nop

nop

nop

nop

mul $r6, $r29, $r6 # Multiply that dimension by the color that we've selected

nop

nop

nop

nop

add $r6, $r6, $r5 # Add that number to the pixel location to get the offset of black

nop

nop

nop

nop

add $r6, $r5, $r1 # Add this to our current position to get our pixel coordinate

nop

nop

nop

nop

addi $r7, $r0, 1 # Start our indexing

nop

nop

nop

nop

addi $r8, $r0, 3 # Choose the line color

nop

nop

nop

nop

beginColorLineThing: bgt $r7, $r5, doneColorLineThing # Draw our 30 pixels then we done

nop

nop

nop

nop

custi1 $r8, $r6, 0 #Store our line color in a pixel

nop

nop

nop

nop

addi $r7, $r7, 1 #Increment our counter

nop

nop

nop

nop

addi $r6, $r6, 1 # Increment our pixel location

nop

nop

nop

nop

j beginColorLineThing

nop

nop

nop

nop

doneColorLineThing:

nop

nop

nop

nop

ret

nop

nop

nop

nop

## End draw selected line

.data

numColors: .word 0x8 #8 colors currently supported ROYGBV + Brown + Black

cursorColor: .word 0x2 #The color index currently being used for the cursor color

pixelMemBegin: .word 0x00010000 # A pointer to the beginning of the pixel memory segment of the program

programMemBegin: .word 0x00001000 #A pointer to the beginning of the program memory segment

maxPixelIndex: .word 0x4b000 # Constant 640\*480 = 307200

numReservedPixels: .word 0x6400 # Constant 640\*40 = 25600 (40 rows)

topFeatureDimension: .word 30 #Dimensions of top feature

colorLineLocation: .word 32 # the location of the color line