.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, 2500 # set the initial pointer position (this position is currently arbitrary)

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

j test

nop

nop

nop

nop

nop

nop

nop

nop

## End test 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, addincthing # 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

addincthing: 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

addi $r2, $r0, 2

nop

nop

nop

nop

addi $r3, $r0, 1

nop

nop

nop

nop

#bne $r3, $r26, noDrawCursor

nop

nop

nop

nop

custi1 $r29, $r25, 0 # fill the location of the current cursor with a color

nop

nop

nop

nop

noDrawCursor: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

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

nop

nop

nop

nop

blt $r29, $r1, addincthing # 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

j checkedInput

nop

nop

nop

nop

addincthing: addi $r29, $r29, 1

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

## End keyboard button checking

.data

numColors: .word 12 #12 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