**Introduction**

This game is called Helicopter, in which the player flies a helicopter through a cave. The player must dodge both the top wall, which randomly changes its downward height, and the bottom wall by increasing and decreasing elevation. For every ten walls the player makes it past score is increased by one. The game is over when the player crashes into either the top or bottom wall.

**Operation Manual**

In this game you play as a helicopter flying through a cave. The helicopter falls down naturally but increases in elevation when the left button on the Basys 3 board is pressed. Be careful however as if you fly too high or too low the helicopter will crash and the game will be over. Your score for each flight is displayed on the seven segment display, directly to the left of the button to gain altitude. When you do crash, simply press the center button on the Basys 3 to try again.

**Peripheral Details (only if new)**

* Given pseudo random number generator
* Given VGA driver for 40 x 30

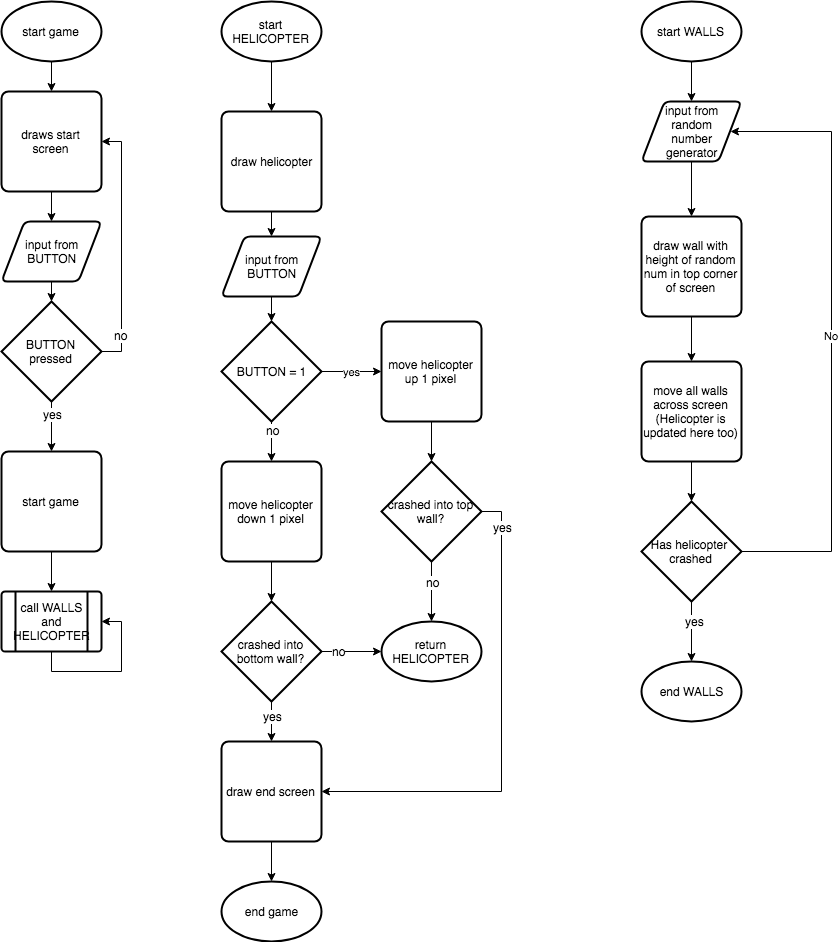
**External Circuit Peripheral**

* None

**Software Design**

The program begins by drawing a start screen and doesn’t begin the actual game until the player presses the button. Once the game starts, the program calls a function to draw the top wall, which height is determined by an input from the random number generator, as well as move it across the screen. This function also calls another function which draws the helicopter and determines its movement. The helicopter’s movement is determined by taking input from the button. If the button is pressed, the program will first check to see if the helicopter will crash into the top wall. If it will crash, the end screen is displayed and the game has ended (until it is reset). If not, the program will change the coordinates of the helicopter to move it up one pixel. If the button is not pressed, the program will do the same, except the check is done with the bottom wall and the helicopter is moved down if it will not crash.

**Flowchart**



**Appendix**

1. Full assembly code listing with comments

; Author: Robert Middleton, Evangeline Barruel, Bridget Benson

; Modifications: bryan mealy

;---------------------------------------------------------------------

CSEG

.ORG 0x10

; outputs

.EQU VGA\_HADD = 0x90

.EQU VGA\_LADD = 0x91

.EQU VGA\_COLOR = 0x92

;inputs

.EQU BTN = 0x42

.EQU SSEG = 0x81

.EQU LEDS = 0x40

.EQU VGA\_IN = 0x93

.EQU RAND = 0x94

.EQU BG\_COLOR = 0x00 ; Background Color: Black

.EQU WALL\_COLOR = 0x1C ; Wall Color: Green

.EQU HELI\_COLOR = 0x03 ; Helicopter Color: Blue

.EQU END\_COLOR = 0xE0 ; End Text Color : Red

.EQU short\_time\_OUTSIDE\_FOR\_COUNT = 0x10

.EQU time\_INSIDE\_FOR\_COUNT = 0x88

.EQU time\_MIDDLE\_FOR\_COUNT = 0x88

.EQU time\_OUTSIDE\_FOR\_COUNT = 0x88

;r0 : X coordinate for heli

;r1 : Y coordinate for heli

;r6 : color

;r7 : Y, temporary for drawing

;r8 : X, temporary for drawing

;r11 : input from the button

;r15 : input from random number generatorIndex of wall array

;r19 : Counter to move walls 10 times

;r23 : counter for score

;r27 : temporary register

;r28 : temporary register

;r29 : temporary register

;r30 : temporary registertemporary register

;---------------------------------------------------------------------

init:

; Reg 8 left side first wall x

; Reg 17 array index

CALL start\_screen ; draws start screen and doesn't

; start game until player hits button

MOV r23, 0x00 ; Set player score to zero

heli:

MOV r1, 0x0F ; starting y coordinate for heli

MOV r0, 0x0A ; x coordinate for heli

MOV r8, 0x00

MOV r6, WALL\_COLOR

;-------------Draws the bottom of the map----------------

bot\_wall:

MOV r7, 0x19

MOV r9, 0x1D

CALL draw\_vertical\_line

SUB r9,0x01

ADD r8,0x01

CMP r8, 0x28

BRNE bot\_wall

MOV r17, 0x30 ; Starting index in scratch for wall array

MOV r20, 0x24 ; Starting x-pos left side of first wall

;------------------Creates and moves first wall 4 times---------------

CALL draw\_wall

CALL short\_pause

CALL short\_pause

CALL move\_wall\_one

CALL short\_pause

CALL move\_wall\_one

CALL short\_pause

CALL move\_wall\_one

CALL short\_pause

CALL move\_wall\_one

CALL short\_pause

;--------------Creates and moves second wall 4 times---------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_two

CALL short\_pause

CALL move\_wall\_two

CALL short\_pause

CALL move\_wall\_two

CALL short\_pause

CALL move\_wall\_two

CALL short\_pause

;------------------Creates and moves third wall 4 times---------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_three

CALL short\_pause

CALL move\_wall\_three

CALL short\_pause

CALL move\_wall\_three

CALL short\_pause

CALL move\_wall\_three

CALL short\_pause

;--------------Creates and moves fourth wall 4 times------------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_four

CALL short\_pause

CALL move\_wall\_four

CALL short\_pause

CALL move\_wall\_four

CALL short\_pause

CALL move\_wall\_four

CALL short\_pause

;--------------Creates and moves fifth wall 4 times--------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_five

CALL short\_pause

CALL move\_wall\_five

CALL short\_pause

CALL move\_wall\_five

CALL short\_pause

CALL move\_wall\_five

CALL short\_pause

;-------------Creates and moves sixth wall 4 times-----------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_six

CALL short\_pause

CALL move\_wall\_six

CALL short\_pause

CALL move\_wall\_six

CALL short\_pause

CALL move\_wall\_six

CALL short\_pause

;----------Creates and moves seventh wall 4 times--------------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_seven

CALL short\_pause

CALL move\_wall\_seven

CALL short\_pause

CALL move\_wall\_seven

CALL short\_pause

CALL move\_wall\_seven

CALL short\_pause

;----------------Creates and moves eighth wall 4 times------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_eight

CALL short\_pause

CALL move\_wall\_eight

CALL short\_pause

CALL move\_wall\_eight

CALL short\_pause

CALL move\_wall\_eight

CALL short\_pause

;-----------------Creates and moves ninth wall 4 times-------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_nine

CALL short\_pause

CALL move\_wall\_nine

CALL short\_pause

CALL move\_wall\_nine

CALL short\_pause

CALL move\_wall\_nine

CALL short\_pause

;--------------Creates and moves tenth wall 4 times------------------

CALL draw\_wall

CALL short\_pause

CALL move\_wall\_ten

CALL short\_pause

CALL move\_wall\_ten

CALL short\_pause

CALL move\_wall\_ten

CALL short\_pause

CALL move\_wall\_ten

CALL short\_pause

;---------------------------------------------------------------------

;- Subroutine: new\_wall\_set

;-

;- This subroutine draws the next set of 10 walls of random heights

;- and updates score

;-

;-

;- Tweaked registers: r17, r19, r20, r23

;---------------------------------------------------------------------

new\_wall\_set:

OUT r23, SSEG

ADD r23, 0x01

MOV r19, 0x00

MOV r17, 0x30

MOV r20, 0x24

Update\_walls: ; Draws a new wall and moves all existing walls

ADD r19, 0x01

CALL draw\_wall

CALL move\_wall\_ten

CALL short\_pause

CALL move\_wall\_ten

CALL short\_pause

CALL move\_wall\_ten

CALL short\_pause

CALL move\_wall\_ten

CALL short\_pause

CMP r19, 0x0A

BRNE update\_walls

BRN new\_wall\_set

main: AND r0, r0 ; nop

BRN main ; continuous loop

draw\_wall: ; Draws a wall from 36 to 40 x-pos at a random height

MOV r7, 0x00

MOV r6, WALL\_COLOR

CALL get\_wall\_height

MOV r8, 0x24

draw\_wall\_inner:

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x01

CMP r8, 0x28

BRNE draw\_wall\_inner

;SUB r8, 0x01

RET

;---------------------------------------------------------------------

;- Subroutines: All move\_wall\_BLANK#

;-

;- These subroutines move the wall associate with the number behind

; the first wall. EX: move\_wall\_three moves the wall 2 spots behind

; the first one.

; Each move wall function moves all walls of a lower number than it

; aswell. EX: move\_wall\_four moves walls four, three, two, and one.

;

; Major difference between each subroutine is the distance behind wall ; one and the index in scratch it calls to remember each walls random

; height.

;- Tweaked registers: r17, r19, r20, r23

;---------------------------------------------------------------------

move\_wall\_one:

MOV r6, BG\_COLOR

CALL draw\_heli ; updates helicopter’s position

MOV r7, 0x00

MOV r8, r20

ADD r8, 0x03 ; THIS LINE IS DIFFERENT IN EACH SUBROUTINE

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x30 ; THIS LINE IS DIFFERENT IN EACH SUBROUTINE

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

SUB r20, 0x01

RET

move\_wall\_two:

CALL move\_wall\_one

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x08

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x31

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_three:

CALL move\_wall\_two

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x0C

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x32

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_four:

CALL move\_wall\_three

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x10

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x33

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_five:

CALL move\_wall\_four

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x14

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x34

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_six:

CALL move\_wall\_five

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x18

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x35

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_seven:

CALL move\_wall\_six

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x1C

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x36

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_eight:

CALL move\_wall\_seven

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x20

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x37

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_nine:

CALL move\_wall\_eight

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x24

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x38

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

move\_wall\_ten:

CALL move\_wall\_nine

MOV r6, BG\_COLOR

MOV r8, r20

ADD r8, 0x28

CALL get\_real\_index

MOV r9, 0x0A

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

MOV r6, WALL\_COLOR

SUB r8, 0x04

LD r9, 0x39

CALL draw\_vertical\_line

MOV r7, 0x00

SUB r9, 0x01

ADD r8, 0x04

RET

Get\_wall\_height: ; Places random wall height into r9 and saves the

; values in scratch array

IN r15, RAND

MOV r27, r15 ; random in as numerator

MOV r28, 0x2A ; 255 / 42 = 6 different wall heights

CALL divide

MOV r31, r30

ST r31, (r17) ; Stores height at current index of scratch

; ram array

ADD r17, 0x01

MOV r9, r30

RET

Get\_real\_index: ; Places mod 40 of r8 back into r8

MOV r27, r8 ; move wall index into numerator

MOV r28, 0x28 ; move 40 into denominator

CALL divide

MOV r8, r27 ; updating index mod 40

RET

;---------------------------------------------------------------------

;- Subroutine: draw\_heli

;-

;- This subroutine draws the helicopter on the display the given coordinates:

;-

;- (X,Y) = (r8,r7) with a color stored in r6

;-

;- and moves it based on input from the button (r11)

;-

;- Tweaked registers: r7

;---------------------------------------------------------------------

draw\_heli:

CALL copy\_coords ; reads in coordinates of helicopter

MOV r6, HELI\_COLOR

CALL draw\_copter ; draw helicopter

MOV r6, BG\_COLOR ; removes copter

CALL draw\_copter

IN r11, BTN ; take input from button

CMP r11, 0x01 ; if input is high, move heli up

; if input is low, move heli down

BRNE heli\_down

heli\_up: CALL top\_collision ; checks to see if heli will crash on top

SUB r7, 0x01 ; move heli up by one pixel

BRN end\_heli

heli\_down: CALL bottom\_collision ; checks to see if heli will crash on bottom

ADD r7, 0x01 ; move heli down by one pixel

end\_heli: MOV r1, r7

RET

;---------------------------------------------------------------------

;- Subroutine: draw\_copter

;-

;- This subroutine draws the helicopter on the display the given coordinates:

;-

;- (X,Y) = (r8,r7) with a color stored in r6

;-

;- Tweaked registers: r7, r8

;---------------------------------------------------------------------

draw\_copter:

CALL copy\_coords ; resets coords to center of heli

draw\_body:

SUB r8, 0x06

MOV r9, r8

ADD r9, 0x09

CALL draw\_horizontal\_line ; draws hor line through center

CALL copy\_coords ; resets coords to center

SUB r7, 0x04

MOV r9, r7

ADD r9, 0x06

CALL draw\_vertical\_line ; draws vert line through center

CALL copy\_coords ; resets coords to center

SUB r7, 0x01

SUB r8, 0x03

MOV r9, r8

ADD r9, 0x05

CALL draw\_horizontal\_line ; draws hor line one above center

ADD r7, 0x02

SUB r8, 0x06

SUB r9, 0x01

CALL draw\_horizontal\_line ; draws hor line one below center

CALL copy\_coords ; resets coords to center

SUB r8, 0x01

SUB r7, 0x02

MOV r9, r7

ADD r9, 0x04

CALL draw\_vertical\_line ; draws vert line one left of center

ADD r8, 0x02

SUB r7, 0x05

SUB r9, 0x01

CALL draw\_vertical\_line ; draws vert line one right of center

ADD r8, 0x01

SUB r7, 0x04

SUB r9, 0x02

CALL draw\_vertical\_line ; draws vert line two right of center

CALL copy\_coords ; resets coords to center

SUB r8, 0x06

SUB r7, 0x01

MOV r9, r7

ADD r9, 0x02

CALL draw\_vertical\_line ; draws back propellor

draw\_long\_prop:

CALL copy\_coords ; resets coords to center

SUB r7, 0x04

SUB r8, 0x04

MOV r9, r8

ADD r9, 0x08

CALL draw\_horizontal\_line ; draws longer top prop

CALL short\_pause

CMP r6, HELI\_COLOR

BRNE draw\_short\_prop

MOV r6, BG\_COLOR

BRN draw\_long\_prop ; erases longer top prop

draw\_short\_prop:

MOV r6, HELI\_COLOR

short\_prop\_loop:

CALL copy\_coords ; resets coords to center

SUB r7, 0x04

SUB r8, 0x02

MOV r9, r8

ADD r9, 0x04

CALL draw\_horizontal\_line ; draws shorter top prop

CMP r6, HELI\_COLOR

BRNE end\_copter

MOV r6, BG\_COLOR

CALL short\_pause

BRN short\_prop\_loop ; erases shorter top prop

end\_copter: CALL copy\_coords ; reset coords to center

RET

;---------------------------------------------------------------------

;- Subroutine: copy\_coords

;-

;- This subroutine copies saved coordinates back into the original registers :

;-

;- X coordinate: r0 => r8

;- Y coordinate: r1 => r7

;---------------------------------------------------------------------

copy\_coords:

MOV r8, r0 ; copies X coordinate

MOV r7, r1 ; copies Y coordinate

RET

;---------------------------------------------------------------------

;- Subroutine: if\_collision

;-

;- This subroutine determines whether the heli at coordinates (r8, r7)

;- has crashed into a wall

;- To see if the heli has crashed into the top wall,

;- compares the color of the top heli pixel (r8, r7 - 4) to the bottom pixel of the top wall

;- To see if the heli has crashed into the bottoom wall,

;- compares the color of top heli pixel (r8, r7 + 2) to the top pixel of the bottom wall

;-

;- VGA color: r6

;---------------------------------------------------------------------

if\_collision:

top\_collision:

CALL copy\_coords ; reads in center coords of heli

SUB r7, 0x05 ; returns pixel above heli

CALL read\_dot ; reads in color of pixel above heli

CMP r6, WALL\_COLOR

BREQ end\_screen ; ends game if pixel is part of the wall

CALL copy\_coords ; resets coords to center of heli

RET

bottom\_collision:

CALL copy\_coords ; reads in center coords of heli

ADD r7, 0x03 ; returns pixel below heli

CALL read\_dot ; reads in color of pixel below heli

CMP r6, WALL\_COLOR

BREQ end\_screen ; ends game if pixel is part of the wall

CALL copy\_coords ; resets coords to center of heli

RET

;--------------------------------------------------------------------

;- Subroutine: divide

;- Parameters:

;- r27 : numerator

;- r28 : denominator

;- Divides r27 by r28. Places whole number quotient in r30 and

; remainder in r27

Tweaked registers: r27, r30

;--------------------------------------------------------------------

divide:

; R30 is counter register

;MOV r27, r15 ; Takes in first input, numerator

;MOV r28, 0x2A ; Takes in second input, denominator

mov r29, 0x00

MOV r30, 0x00

CMP r28, 0x00 ; Checking to see if denominator is zero

BREQ exactWholeNumEnd ; Skip to end if divide by zero

findWholeNumberQuotient:

CMP r27, r29 ; Comparing numerator and multiple of denominator

BRCS endWholeNumLoop ; End loop if multiple of denominator > numerator

BREQ exactWholeNumEnd

ADD r29, r28 ; Multiplying denominator by itself

CMP R29, r28 ; If the multiple of denominator is < original

; denominator then whole number multiple found

BRCS endWholeNumLoop

ADD r30, 0x01 ; Adding one to counter of denominator self

; multiplies

BRN findWholeNumberQuotient

endWholeNumLoop:

SUB R29, R28 ; Subtract last denominator Multiply to get largest

; less than numerator

SUB R30, 0x01 ; Subtract 1 from counter to get accurate number

; of loops

exactWholeNumEnd:

SUB R27, R29 ; Subtracting largest whole number multiple

; of denominator less than numerator from numerator

; to find remainder

ADD r30, 0x04 ; Min wall height is 4

RET

; R30 holds whole number

; R27 holds remainder

;---------------------------------------------------------------------

;- Subroutine: start\_screen

;-

;- This subroutine displays a start screen until the player presses the button

;-

;- (to write START)

;- X coordinate: r8

;- Y coordinate: r7

;- draw color: r6

;---------------------------------------------------------------------

start\_screen:

CALL draw\_background

MOV r6, WALL\_COLOR ; changes text color

start\_draw\_S:

MOV r7, 0x0B

MOV r8, 0x08

MOV r9, 0x0B

CALL draw\_horizontal\_line ; draws top h line of S

MOV r8, 0x08

MOV r9, 0x0E

CALL draw\_vertical\_line ; draws top left v line of S

MOV r7, 0x0E

MOV r8, 0x08

MOV r9, 0x0B

CALL draw\_horizontal\_line ; draws center h line of S

MOV r8, 0x0B

MOV r9, 0x11

CALL draw\_vertical\_line ; draws bottom right v line of S

MOV r7, 0x11

MOV r8, 0x08

MOV r9, 0x0B

CALL draw\_horizontal\_line ; draws bottom h line of S

MOV r8, 0x0B

MOV r7, 0x0C

CALL draw\_dot ; draws top right dot of S (for curl)

MOV r8, 0x08

MOV r7, 0x10

CALL draw\_dot ; draws bottom left dot of S

start\_draw\_T\_1:

MOV r8, 0x0D

MOV r7, 0x0B

MOV r9, 0x10

CALL draw\_horizontal\_line ; draws top h line of T

MOV r8, 0x0D

MOV r7, 0x0C

MOV r9, 0x10

CALL draw\_horizontal\_line ; draws second h line of T

MOV r8, 0x0E

MOV r7, 0x0D

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of T

MOV r8, 0x0F

MOV r7, 0x0D

MOV r9, 0x11

CALL draw\_vertical\_line ; draws right v line of T

start\_draw\_A:

MOV r8, 0x12

MOV r7, 0x0C

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of A

MOV r8, 0x15

MOV r7, 0x0C

MOV r9, 0x11

CALL draw\_vertical\_line ; draws right v line of A

MOV r8, 0x13

MOV r7, 0x0B

MOV r9, 0x14

CALL draw\_horizontal\_line ; draws top h line of A

MOV r8, 0x13

MOV r7, 0x0E

MOV r9, 0x14

CALL draw\_horizontal\_line ; draws center h line of A

start\_draw\_R:

MOV r8, 0x17

MOV r7, 0x0B

MOV r9, 0x19

CALL draw\_horizontal\_line ; draws top h line of R

MOV r8, 0x17

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of R

MOV r7, 0x0E

MOV r9, 0x19

CALL draw\_horizontal\_line ; draws center h line of R

MOV r8, 0x1A

MOV r7, 0x0C

MOV r9, 0x11

CALL draw\_vertical\_line ; draws right v line of R

MOV r6, BG\_COLOR

MOV r7, 0x0E

CALL draw\_dot ; erases center dot of right v line of R

start\_draw\_T\_2:

MOV r6, WALL\_COLOR

MOV r8, 0x1C

MOV r7, 0x0B

MOV r9, 0x1F

CALL draw\_horizontal\_line ; draws top h line of T

MOV r8, 0x1C

MOV r7, 0x0C

MOV r9, 0x1F

CALL draw\_horizontal\_line ; draws second h line of T

MOV r8, 0x1D

MOV r7, 0x0D

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of T

MOV r8, 0x1E

MOV r7, 0x0D

MOV r9, 0x11

CALL draw\_vertical\_line ; draws right v line of T

start\_wait:

IN r11, BTN ; input from button

CMP r11, 0x01

BRNE start\_wait ; only start game when button is pressed

CALL draw\_background ; resets screen

RET

;---------------------------------------------------------------------

;- Subroutine: end\_screen

;-

;- This subroutine displays an end screen when the player crashes

;-

;- (to write DEAD)

;- X coordinate: r8

;- Y coordinate: r7

;- draw color: r6

;---------------------------------------------------------------------

end\_screen:

CALL draw\_background

MOV r6, END\_COLOR ; changes text color

end\_draw\_D\_1:

MOV r8, 0x09

MOV r7, 0x0B

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of D

MOV r7, 0x0B

MOV r9, 0x0B

CALL draw\_horizontal\_line ; draws top h line of D

MOV r8, 0x0A

MOV r7, 0x11

MOV r9, 0x0B

CALL draw\_horizontal\_line ; draws bottom h line of D

MOV r8, 0x0C

MOV r7, 0x0C

MOV r9, 0x10

CALL draw\_vertical\_line ; draws right v line of D

end\_draw\_E:

MOV r8, 0x0F

MOV r7, 0x0B

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of E

MOV r7, 0x0B

MOV r9, 0x12

CALL draw\_horizontal\_line ; draws top h line of E

MOV r8, 0x10

MOV r7, 0x0E

MOV R9, 0x11

CALL draw\_horizontal\_line ; draws center h line of E

MOV r8, 0x10

MOV r7, 0x11

MOV r9, 0x12

CALL draw\_horizontal\_line ; draws bottom h line of E

end\_draw\_A:

MOV r8, 0x15

MOV r7, 0x0C

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of A

MOV r8, 0x18

MOV r7, 0x0C

MOV r9, 0x11

CALL draw\_vertical\_line ; draws right v line of A

MOV r8, 0x16

MOV r7, 0x0B

MOV r9, 0x17

CALL draw\_horizontal\_line ; draws top h line of A

MOV r8, 0x16

MOV r7, 0x0E

MOV r9, 0x17

CALL draw\_horizontal\_line ; draws center h line of A

end\_draw\_D\_2:

MOV r8, 0x1B

MOV r7, 0x0B

MOV r9, 0x11

CALL draw\_vertical\_line ; draws left v line of D

MOV r7, 0x0B

MOV r9, 0x1D

CALL draw\_horizontal\_line ; draws top h line of D

MOV r8, 0x1C

MOV r7, 0x11

MOV r9, 0x1D

CALL draw\_horizontal\_line ; draws bottom h line of D

MOV r8, 0x1E

MOV r7, 0x0C

MOV r9, 0x10

CALL draw\_vertical\_line ; draws right v line of D

BRN main

pause: ; PAUSE OF DURATION .1 SECOND

MOV R29, time\_OUTSIDE\_FOR\_COUNT ;set outside for loop count

outside\_for:

SUB R29, 0x01

MOV R28, time\_MIDDLE\_FOR\_COUNT ;set middle for loop count

middle\_for:

SUB R28, 0x01

MOV R27, time\_INSIDE\_FOR\_COUNT ;set inside for loop count

inside\_for:

SUB R27, 0x01

BRNE inside\_for

OR R28, 0x00 ;load flags for middle for counter

BRNE middle\_for

OR R29, 0x00 ;load flags for outsde for counter value

BRNE outside\_for

RET

short\_pause: ; A SMALL PAUSE USED FOR GRAPHIC UPDATES

MOV R29, short\_time\_OUTSIDE\_FOR\_COUNT ;set outside for loop count

s\_outside\_for:

SUB R29, 0x01

MOV R28, time\_MIDDLE\_FOR\_COUNT ;set middle for loop count

s\_middle\_for:

SUB R28, 0x01

MOV R27, time\_INSIDE\_FOR\_COUNT ;set inside for loop count

s\_inside\_for:

SUB R27, 0x01

BRNE s\_inside\_for

OR R28, 0x00 ;load flags for middle for counter

BRNE s\_middle\_for

OR R29, 0x00 ;load flags for outsde for counter value

BRNE s\_outside\_for

RET

;--------------------------------------------------------------------

;- Subroutine: draw\_horizontal\_line

;-

;- Draws a horizontal line from (r8,r7) to (r9,r7) using color in r6

;-

;- Parameters:

;- r8 = starting x-coordinate

;- r7 = y-coordinate

;- r9 = ending x-coordinate

;- r6 = color used for line

;-

;- Tweaked registers: r8, r9

;--------------------------------------------------------------------

draw\_horizontal\_line:

ADD r9,0x01 ; go from r8 to r9 inclusive

draw\_horiz1:

CALL draw\_dot ; draws dots from r8 to r9

ADD r8,0x01

CMP r8,r9

BRNE draw\_horiz1

RET

;--------------------------------------------------------------------

;---------------------------------------------------------------------

;- Subroutine: draw\_vertical\_line

;-

;- Draws a horizontal line from (r8,r7) to (r8,r9) using color in r6

;-

;- Parameters:

;- r8 = x-coordinate

;- r7 = starting y-coordinate

;- r9 = ending y-coordinate

;- r6 = color used for line

;-

;- Tweaked registers: r7, r9

;--------------------------------------------------------------------

draw\_vertical\_line:

ADD r9,0x01 ; go from r7 to r9 inclusive

draw\_vert1:

CALL draw\_dot ; draws dots from r7 to r9

ADD r7,0x01

CMP r7,R9

BRNE draw\_vert1

RET

;--------------------------------------------------------------------

;---------------------------------------------------------------------

;- Subroutine: draw\_background

;-

;- Fills the 30x40 grid with one color using successive calls to

;- draw\_horizontal\_line subroutine.

;-

;- Tweaked registers: r13, r7, r8, r9

;----------------------------------------------------------------------

draw\_background:

MOV r6,BG\_COLOR ; use default color

MOV r13,0x00 ; r13 keeps track of rows

start: MOV r7,r13 ; load current row count

MOV r8,0x00 ; restart x coordinates

MOV r9,0x27

CALL draw\_horizontal\_line

ADD r13,0x01 ; increment row count

CMP r13,0x1D ; see if more rows to draw

BRNE start ; branch to draw more rows

RET

;---------------------------------------------------------------------

;---------------------------------------------------------------------

;- Subrountine: draw\_dot

;-

;- This subroutine draws a dot on the display the given coordinates:

;-

;- (X,Y) = (r8,r7) with a color stored in r6

;-

;- Tweaked registers: r4, r5

;---------------------------------------------------------------------

draw\_dot:

MOV r4,r7 ; copy Y coordinate

MOV r5,r8 ; copy X coordinate

AND r5,0x3F ; make sure top 2 bits cleared

AND r4,0x1F ; make sure top 3 bits cleared

LSR r4 ; need to get the bot 2 bits of r4 into sA

BRCS dd\_add40

t1: LSR r4

BRCS dd\_add80

dd\_out: OUT r5,VGA\_LADD ; write bot 8 address bits to register

OUT r4,VGA\_HADD ; write top 3 address bits to register

OUT r6,VGA\_COLOR ; write data to frame buffer

RET

dd\_add40: OR r5,0x40 ; set bit if needed

CLC ; freshen bit

BRN t1

dd\_add80: OR r5,0x80 ; set bit if needed

BRN dd\_out

; --------------------------------------------------------------------

;---------------------------------------------------------------------

;- Subrountine: read\_dot

;-

;- This subroutine reads a dot on the display the given coordinates:

;-

;- (X,Y) = (r8,r7) with a color stored in r6

;-

;- Tweaked registers: r4, r5

;---------------------------------------------------------------------

read\_dot:

MOV r4,r7 ; copy Y coordinate

MOV r5,r8 ; copy X coordinate

AND r5,0x3F ; make sure top 2 bits cleared

AND r4,0x1F ; make sure top 3 bits cleared

LSR r4 ; need to get the bot 2 bits of r4 into sA

BRCS r\_dd\_add40

r\_t1: LSR r4

BRCS r\_dd\_add80

r\_dd\_out: OUT r5,VGA\_LADD ; write bot 8 address bits to register

OUT r4,VGA\_HADD ; write top 3 address bits to register

IN r6,VGA\_IN ; write data to frame buffer

RET

r\_dd\_add40:OR r5,0x40 ; set bit if needed

CLC ; freshen bit

BRN r\_t1

r\_dd\_add80: OR r5,0x80 ; set bit if needed

BRN r\_dd\_out

; --------------------------------------------------------------------

end:

2. Peripheral VHDL code listing

* No new VHDL code