David Caldera
CPE 233
3/14/2022

Design Project: Obstacle Course

Introduction:

The device built is an obstacle course game that also allows for added development of maze-like game boards with multiple possible end goals. The game build for this demonstration only features 1 level uses a reset to reset the game board once reaching the game board. The player navigates a game board, with 4 directional controls, colored blue with barriers colored red that the player cannot traverse. The player navigates the game board until they reach the end goal represented as a green block on the game board. The device checks for keyboard detection runs at the BASYS 3 100 MHz clock, interrupts will be detected on a positive edge of the clock is. This keyboard input is used for interrupt detection, and when an interrupt occurs we read the scan code coming from the keystroke and save into the address register of the keyboard, then load that keystroke data to into the next clock cycle to cause interrupt to happen. This game only uses A, W, S, D controls for 4 directional movement; other keystrokes will be detected but since they are not mapped for controls, they will have no effect player object on game board. More maps can be designed for this device and can inserted into software by just recoloring board and having a separate sub-routine where object placement will differ from past gameboard.

Operation Manual:

Gameboard:

The game is an obstacle course for the player to navigate such as the one below in figure 1. The player will navigate the gameboard as white block on the display located below labeled as the start section of the game in figure 1. Any spot on the gameboard that is not a red colored obstacle can be traversed across, and in the case of the green block located in the top right corner of the gameboard it will trigger an end game state as the player has completed the maze. User uses four directional control to navigate the gameboard and only must operate AWSD control scheme with reset button once end goal has been reached.

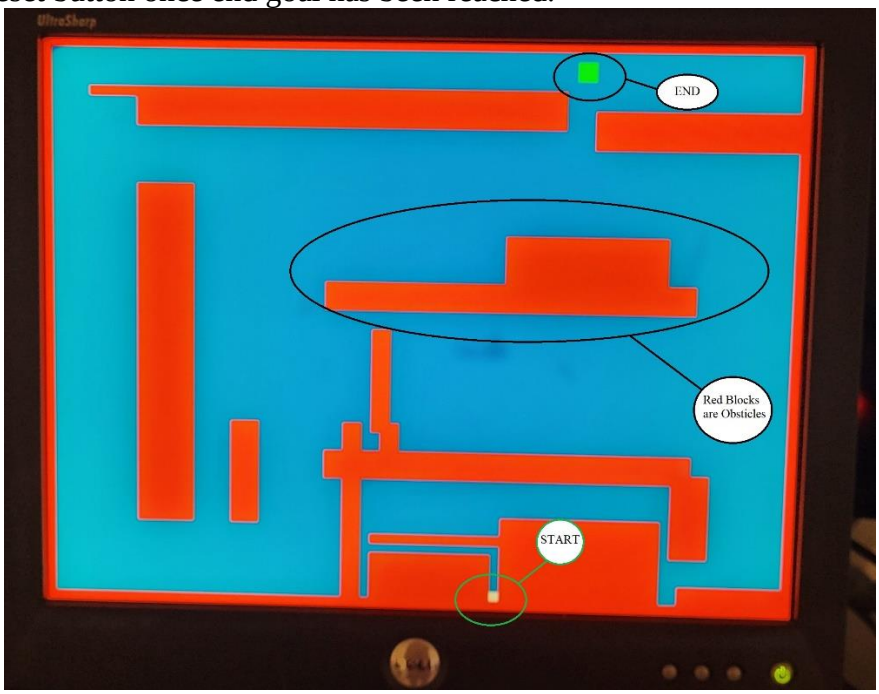

Figure 1: Example Gameboard

User Control:

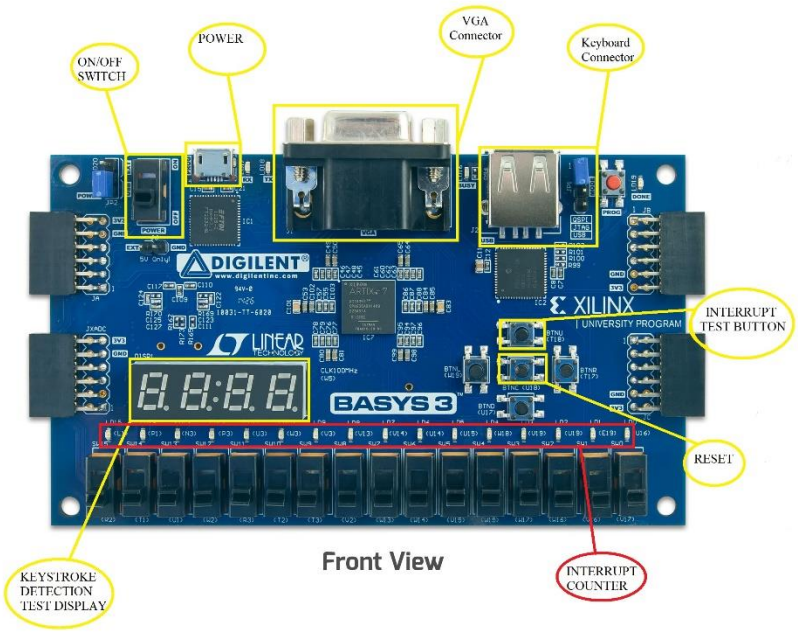

Figure 2: BASYS 3 Board configuration

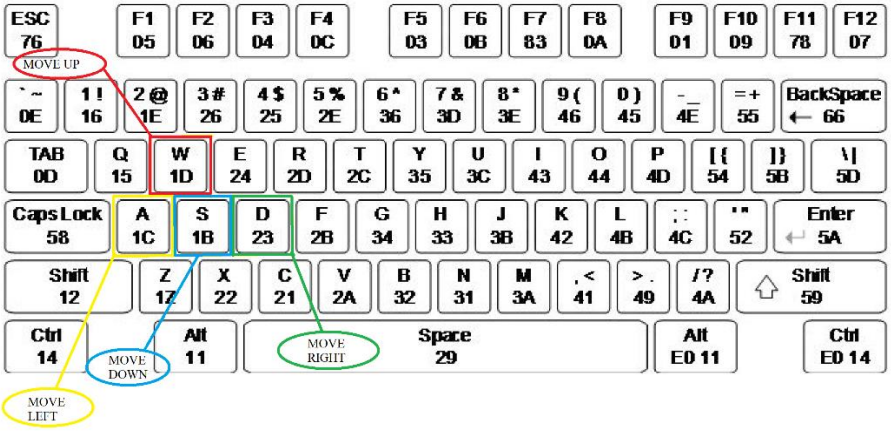

Figure 3: Keyboard Mapping

To operate device user will have to first plug in power source the section labeled “Power” in figure 2 and put the “ON/OFF” switch to ON. Following this board will need to be programed with provided software and hardware. User will have to connect board to VGA display and connect a keyboard to play the game.

Once game is hooked up user can test to see keyboard is compatible with hardware by pressing any keyboard button and observing the 7-Segment display being used a Keystroke Detection Test Display in Figure 2. If keyboard button input is not displaying corresponding values from figure 3 but is counting interrupts on the interrupt counter shown in figure 2 make sure that the

keyboard plugged in is a tenkey keyboard and not a tenkeyless keyboard, as this will cause interfacing issues.

The game will be played by used AWSD controls as shown in figure 3, meaning that A will move player to the left, W will move player up, S will move player down, and D will move player to the right. Once player has reached the green block in figure 1 the display will turn completely green to signify that the player has reached the end of the course. The player will need to push the reset button on the BASYS 3 board as shown on figure 2 to restart the gameboard play again.

Software Design:

In figure 4 we can see the main diagram that contains all of the top level subroutine calls as well as the draw_background subroutine call that draws out the background of the game board. For the main flow chart, we can see that we first set up our prepheral addresses for our VGA output and inputs, we initialize our stack pointer, set our scancode pointer for the keyboard input, we also create our color detection variables, and create our interrupt handler with all of its corresponding parts. After clearing the 7-Segment display we enter our first subroutine call draw_background. We then clear the register t4 to zero and enter DrawCourse subroutine to draw our obstacles onto the gameboard. Entering the game subroutine, the player now can interact with the game and is able to reach the end subroutine which happens upon completion of the game.

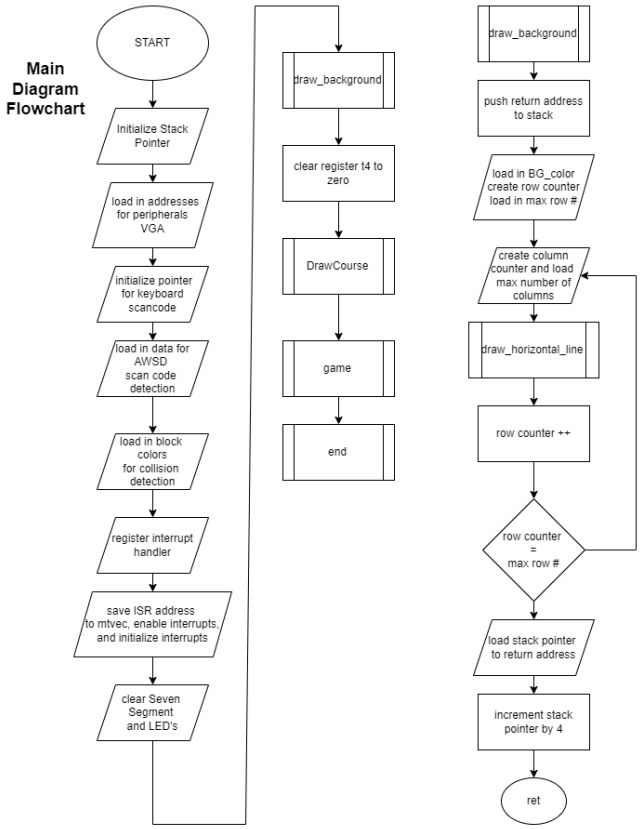

Figure 4: Flow Chart Part 1

Referring to figure 4; In draw_background subroutine we utilize a stack push to save our return address at that location. We then use color data, row and column boundaries, and the draw_horizontal_line subroutine to draw multiple horizontal lines that stretch to the end of the screen. Until we reach the final row position required to paint screen the background color we will continue looping. Upon matching the max number of rows, we pop out stack value out, the return address, and return to our call location.

In figure 5 we can see the DrawCourse subroutine shown in figure 4; DrawCourse generates the obstacles for the gameboard. The subroutine houses multiple calls to draw_dot, draw_horizontal_line, and draw_vertical_line subroutine, so to shorten the flow chart it is represented as load containing the general course of actions that occur in the subroutine. Depending on the subroutine call it will call either of the three to generate obstacles and upon finishing block generation it will return to jump into next subroutine shown in figure 4.

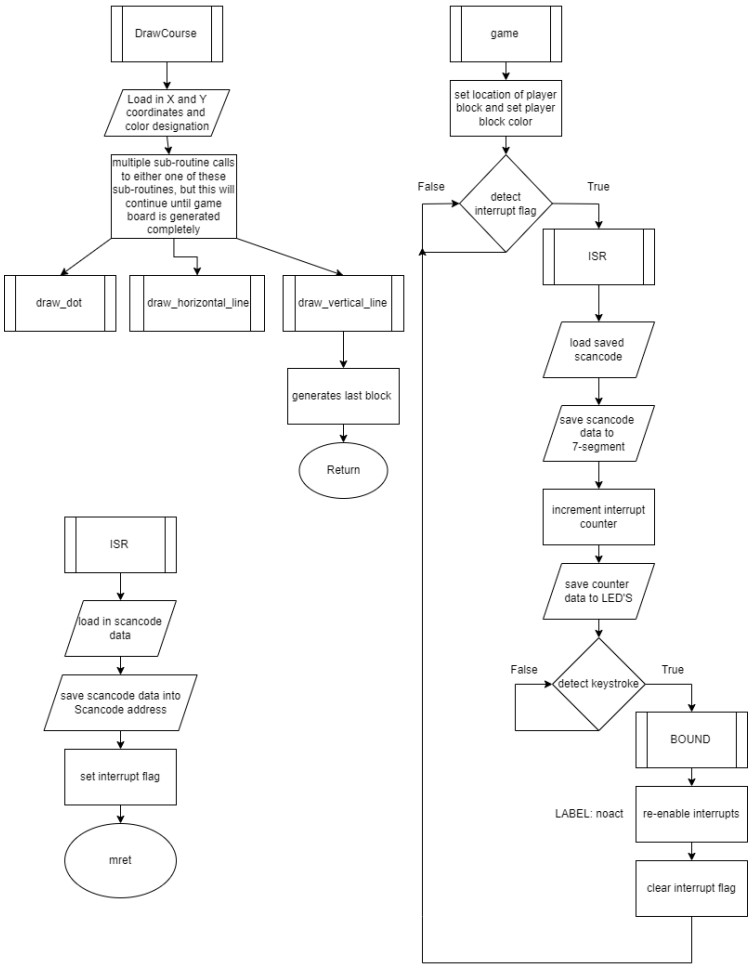

Figure 5: Flow Chart Part 2

The game subroutine shown in figure 5 will set up our main function in which it will continue looping through its detection code until a keystroke from the user is detected. First it sets the location of the player block on the board to a default case and will set player block color. Upon keyboard detection it will cause an interrupt state and will jump into the ISR to complete

its interrupt commands. It will load in the scancode data from the keyboard and save it to the scancode address register, then will set the interrupt flag and return.

Once returned we will load the saved scancode and write it to the 7-segment display. The interrupt counter will be incremented and saved to the LED's. The interrupt flag will be used to enter the BOUND subroutine, which will check to see if the keyboard input is one that is mapped for movement and if there will be a collision with a obstacle. Exiting out of the Bound subroutine we will re-enable interrupts and clear the interrupt flag before returning to the interrupt detection.

In figure 6 we examine the BOUND, Astroke, and Wstroke subroutines. The BOUND subroutine will use the scancode taken from the keyboard input and check to see if it either A, W, S, or D. If it is any of the mapped keystrokes it will enter it's respective subroutine, and if it is a keystroke that is not mapped it will jump the noact label, no action taken, located in the game subroutine.

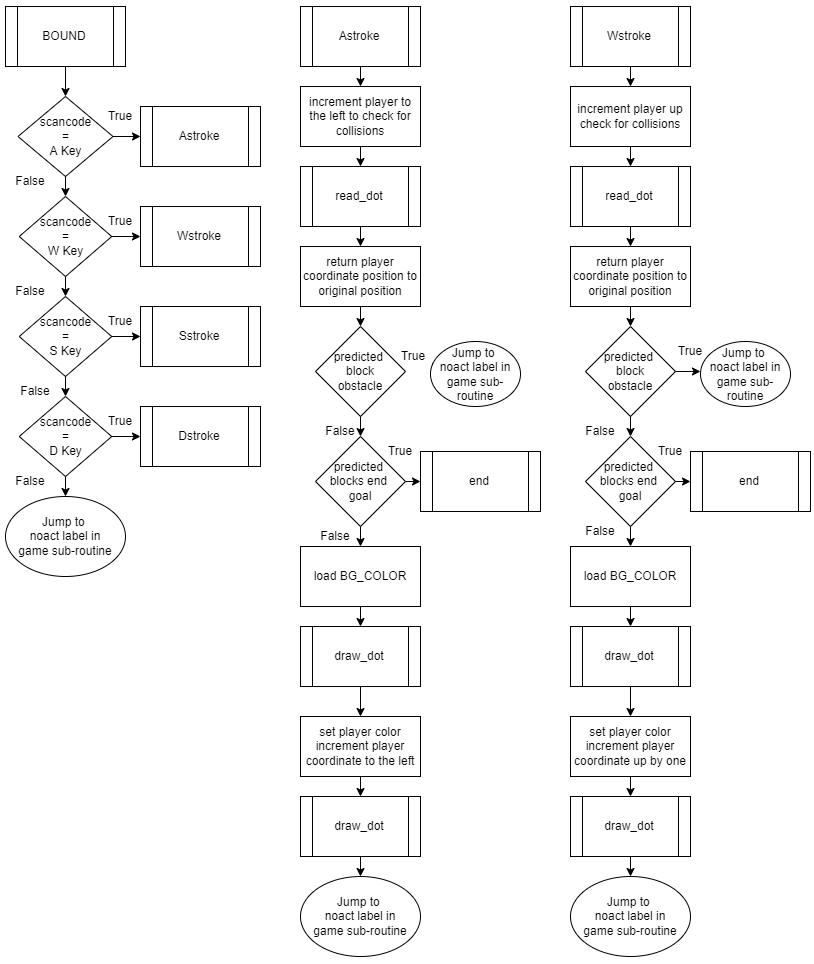

Figure 6: Flow Chart Part 3

The Astroke subroutine will be similar to all other movement-based subroutines but will differ in what registers are used for their coordinate adjustments. The Astroke subroutine will first check for obstacle collisions used the read_dot subroutine. If an obstacle collion is detection is detected will jump to the noact label, but if a green block collision is detected then we have

reached the end goal and will jump the end subroutine. If no collision is detected, then we will color the present location of the block the background color and color the future position of the block the player object color emulate movement.

The Wstroke subroutine will function in the same way as the Astroke subroutine but will instead move vertically instead horizontal movement and like wise check in the same fashion. This is same for Sstroke subroutine, shown in figure 7, as it will be vertically down instead of up, but again will function in the same way as Wstroke or Astroke. Dstroke, shown in figure 7, will function similar to Astroke as well, but will instead check for horizontal collision and move player to the right instead of the left

The draw_horizontal_line subroutine, shown in figure7, will draw a horizontal line onto the gameboard and will do this through drawing multiple dots for a designated starting column and end column as well as the row to draw onto. Similar to the draw background function will decrement stack to save our return address and pop it out in order to return.

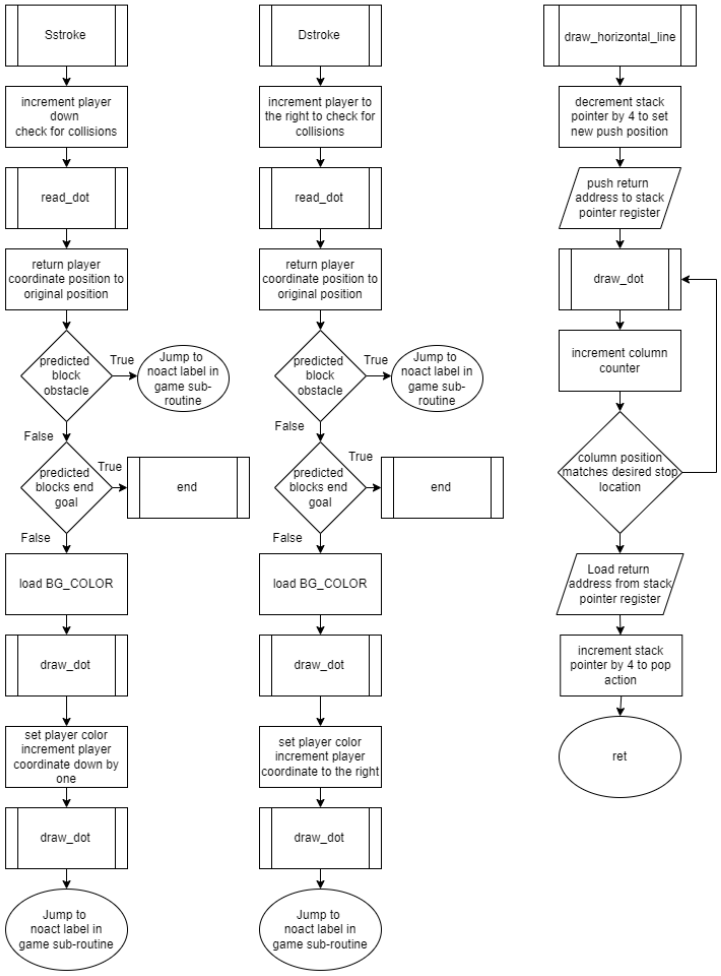

Figure 7: Flow Chart Part 4

In figure 8 we will examine the last subroutines draw_vertical_line, draw_dot, read_dot, end, and draw_endground. The draw_vertical_line will function very similar to the draw_horizontal_line subroutine shown in figure 7, but will instead have a starting row location, end row location, and a column location and color designation also.

The draw_dot subroutine will draw a dot at a designated column and row location with a designated color. It will combine the two coordinates and write that that location the gameboard then color it using the designated color data given to it.

The read_dot subroutine is extremely similar to the draw_dot subroutine and in fact only features 1 change which is to read the color data at a given coordinate and save that data to a register in order to be used for obstacle or end goal detection.

We finally reach our end subroutine which will loop infinitely into the draw_endground subroutine creating a constant end screen to be displayed to the VGA device. In the draw_endground subroutine it will work similar to the draw_background subroutine in that instead coloring the screen as the background color it will instead color the entire green instead to signify that the player reached the goal.

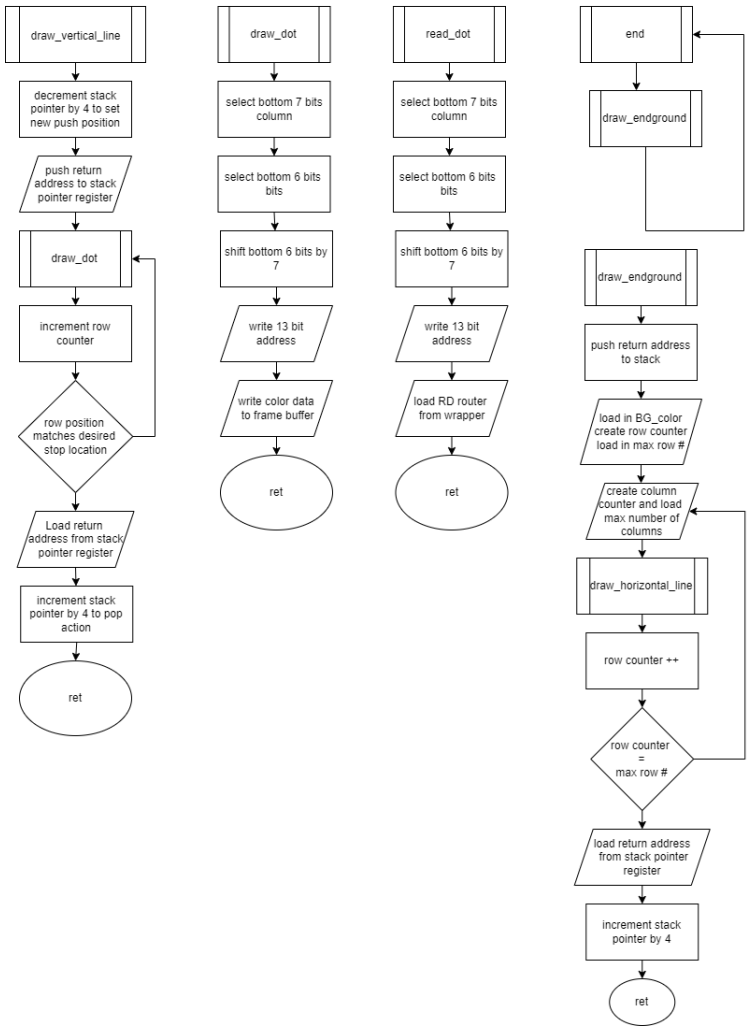

Figure 8: Flow Chart Part 5

Appendix:

1. Full Assembly Code Listing

```
# This program will fill the screen with a BG_COLOR before
# drawing 3 dots, a horizontal, and vertical lines
```

```
#
# coordinates are given in row major format
# (col,row) = (x,y)
# written by J. Calllenes and P. Hummel

.eqv BG_COLOR, 0x0F      # light blue (0/7 red, 3/7 green, 3/3 blue)
.eqv VG_ADDR, 0x11000120
.eqv VG_COLOR, 0x11000140
.eqv VG_READ, 0x11000160
.eqv MMIO,0x11000000

.data
SCANCODE:
.text

init:
        li sp, 0x10000     #initialize stack pointer
      li s2, VG_ADDR     #load MMIO addresses
      li s3, VG_COLOR
      li s11, VG_READ      #  READ from Buffer

      li   s0, MMIO        # pointer for MMIO
            la   s1, SCANCODE    # pointer to scancode

            li s5, 28        # A keystroke
            li s6, 29        # W keystroke
            li s7, 27        # S keystroke
            li s8, 35        # D keystroke

            li s9, 0xE0          # red
            li s10, 0x1C           # green

            la t3, ISR         # register the interrupt handler
            csrrw x0, mtvec, t3
            li    t3, 1           # enable interrupts
            csrrw x0, mie, t3

            add   t3, x0, x0      # initialize  flag
            add   s4, x0, x0      # initialize interrupt count
            sw    s4, 0x40(s0)    # clear 7Seg
            sw    s4, 0x20(s0)    # clear LEDs


      # fill screen using default color
      call  draw_background  # must not modify s2, s3
      add t4, x0, x0          # t4 to zero again
      j DrawCourse

startgame:
      j game
```

```
DrawCourse:
      li a0, 57         # X coordinate
      li a1, 2          # Y coordinate
      li a3, 0x1C       # color green (7/7 red, 0/7 green, 0/3 blue)
      call draw_dot  # must not modify s2, s3

      li a0, 56         # X coordinate
      li a1, 2          # Y coordinate
      li a3, 0x1C       # color green (7/7 red, 0/7 green, 0/3 blue)
      call draw_dot  # must not modify s2, s3

      li a0, 57         # X coordinate
      li a1, 3          # Y coordinate
      li a3, 0x1C       # color green (7/7 red, 0/7 green, 0/3 blue)
      call draw_dot  # must not modify s2, s3

      li a0, 56         # X coordinate
      li a1, 3          # Y coordinate
      li a3, 0x1C       # color green (7/7 red, 0/7 green, 0/3 blue)
      call draw_dot  # must not modify s2, s3

      # top screen border
      li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
      li a0, 0          # start X coordinate
      li a1, 0          # Y coordinate
      li a2, 79         # ending X coordinate
      call draw_horizontal_line  # must not modify: a3, s2, s3
      # low screen border
      li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
      li a0, 0          # start X coordinate
      li a1, 59         # Y coordinate
      li a2, 79         # ending X coordinate
      call draw_horizontal_line  # must not modify: a3, s2, s3
      # left side border
      li a0, 0          # X coordinate
      li a1, 1          # starting Y coordinate
      li a2, 59         # ending Y coordinate
      call draw_vertical_line  # must not modify s2, s3
      # right side border
      li a0, 79         # X coordinate
      li a1, 0          # starting Y coordinate
      li a2, 59         # ending Y coordinate
      call draw_vertical_line  # must not modify s2, s3

      # player block
      li a0, 48         # X coordinate
      li a1, 58         # Y coordinate
      li a3, 0xFF       # color red (7/7 red, 0/7 green, 0/3 blue)
      call draw_dot  # must not modify s2, s3
```

```
# BLOCK 1
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 35         # start X coordinate
li a1, 54         # Y coordinate
li a2, 47         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 35         # start X coordinate
li a1, 55         # Y coordinate
li a2, 47         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 35         # start X coordinate
li a1, 56         # Y coordinate
li a2, 47         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 35         # start X coordinate
li a1, 57         # Y coordinate
li a2, 47         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 35         # start X coordinate
li a1, 58         # Y coordinate
li a2, 47         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 35         # start X coordinate
li a1, 59         # Y coordinate
li a2, 47         # ending X coordinate
call draw_horizontal_line

# BLOCK 2
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 54         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 55         # Y coordinate
li a2, 65         # ending X coordinate
```

```
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 56         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 57         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 58         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 59         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

# BLOCK 3
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 50         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 51         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 52         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 53         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line
```

```
# BLOCK 4
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 20         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 21         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 22         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 23         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 49         # start X coordinate
li a1, 24         # Y coordinate
li a2, 65         # ending X coordinate
call draw_horizontal_line

# BLOCK 5
li a0, 68                 # X coordinate
li a1, 45         # starting Y coordinate
li a2, 53   # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 67                 # X coordinate
li a1, 45         # starting Y coordinate
li a2, 53   # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 68                 # X coordinate
li a1, 45         # starting Y coordinate
li a2, 53   # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 69                 # X coordinate
li a1, 45         # starting Y coordinate
```

```
li a2, 53   # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 70                # X coordinate
li a1, 45         # starting Y coordinate
li a2, 53   # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

# BLOCK 6
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 68         # start X coordinate
li a1, 57         # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 68         # start X coordinate
li a1, 58         # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 68         # start X coordinate
li a1, 58         # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

# BLOCK 7
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 30         # start X coordinate
li a1, 25         # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 30         # start X coordinate
li a1, 26         # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 30         # start X coordinate
li a1, 27         # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

# BLOCK 8
li a0, 20                # X coordinate
li a1, 41         # starting Y coordinate
li a2, 50         # ending Y coordinate
```

```
call draw_vertical_line  # must not modify s2, s3

li a0, 21                       # X coordinate
li a1, 41         # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

# BLOCK 9
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 5          # start X coordinate
li a1, 5          # Y coordinate
li a2, 20         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 35         # start X coordinate
li a1, 52         # Y coordinate
li a2, 50         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

# BLOCK 10
li a0, 32                 # X coordinate
li a1, 40         # starting Y coordinate
li a2, 59         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 33                 # X coordinate
li a1, 40         # starting Y coordinate
li a2, 59         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3


# BLOCK 11
li a0, 10                 # X coordinate
li a1, 15         # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 11                 # X coordinate
li a1, 15         # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 12                 # X coordinate
li a1, 15         # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 13                 # X coordinate
li a1, 15         # starting Y coordinate
```

```
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 14                 # X coordinate
li a1, 15         # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 15                 # X coordinate
li a1, 15         # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

# BLOCK 12
li a0, 20                 # X coordinate
li a1, 9          # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 21                 # X coordinate
li a1, 9          # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 22                 # X coordinate
li a1, 9          # starting Y coordinate
li a2, 50         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

# BLOCK 15
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 10         # start X coordinate
li a1, 5          # Y coordinate
li a2, 54         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 10         # start X coordinate
li a1, 6          # Y coordinate
li a2, 54         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 10         # start X coordinate
li a1, 7          # Y coordinate
li a2, 54         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 10         # start X coordinate
```

```
li a1, 8          # Y coordinate
li a2, 54         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

# BLOCK 19
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 58         # start X coordinate
li a1, 7          # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 58         # start X coordinate
li a1, 8          # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 58         # start X coordinate
li a1, 9          # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 58         # start X coordinate
li a1, 10         # Y coordinate
li a2, 79         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

# BLOCK 20
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 30         # start X coordinate
li a1, 43         # Y coordinate
li a2, 68         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 30         # start X coordinate
li a1, 44         # Y coordinate
li a2, 68         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 30         # start X coordinate
li a1, 45         # Y coordinate
li a2, 68         # ending X coordinate
call draw_horizontal_line  # must not modify: a3, s2, s3

# BLOCK 21
li a0, 35                 # X coordinate
```

```
li a1, 30         # starting Y coordinate
li a2, 40         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 36                 # X coordinate
li a1, 30         # starting Y coordinate
li a2, 42         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 37                 # X coordinate
li a1, 40         # starting Y coordinate
li a2, 43         # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

# BLOCK 22
li a0, 55                 # X coordinate
li a1, 5          # starting Y coordinate
li a2, 8          # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 56                 # X coordinate
li a1, 5          # starting Y coordinate
li a2, 8          # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

li a0, 57                 # X coordinate
li a1, 5          # starting Y coordinate
li a2, 8          # ending Y coordinate
call draw_vertical_line  # must not modify s2, s3

# BLOCK 23
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 1          # start X coordinate
li a1, 15         # Y coordinate
li a2, 9          # ending X coordinate
call draw_horizontal_line

li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 1          # start X coordinate
li a1, 16         # Y coordinate
li a2, 9          # ending X coordinate
call draw_horizontal_line
li a3, 0xE0       # color red (7/7 red, 0/7 green, 0/3 blue)
li a0, 1          # start X coordinate
li a1, 17         # Y coordinate
li a2, 9          # ending X coordinate
call draw_horizontal_line

# BLOCK 24
li a0, 13           # X coordinate
```

```
	li a1, 55         # starting Y coordinate
	li a2, 57         # ending Y coordinate
	call draw_vertical_line  # must not modify s2, s3

	li a0, 14                 # X coordinate
	li a1, 55         # starting Y coordinate
	li a2, 57         # ending Y coordinate
	call draw_vertical_line  # must not modify s2, s3

	li a0, 15                 # X coordinate
	li a1, 55         # starting Y coordinate
	li a2, 57         # ending Y coordinate
	call draw_vertical_line  # must not modify s2, s3

	j startgame # continuous loop

# draws a horizontal line from (a0,a1) to (a2,a1) using color in a3
# Modifies (directly or indirectly): t0, t1, a0, a2
draw_horizontal_line:
	addi sp,sp,-4
	sw ra, 0(sp)
	addi a2,a2,1	#go from a0 to a2 inclusive
draw_horiz1:
	call draw_dot  # must not modify: a0, a1, a2, a3
	addi a0,a0,1
	bne a0,a2, draw_horiz1
	lw ra, 0(sp)
	addi sp,sp,4
	ret

# draws a vertical line from (a0,a1) to (a0,a2) using color in a3
# Modifies (directly or indirectly): t0, t1, a1, a2
draw_vertical_line:
	addi sp,sp,-4
	sw ra, 0(sp)
	addi a2,a2,1
draw_vert1:
	call draw_dot  # must not modify: a0, a1, a2, a3
	addi a1,a1,1
	bne a1,a2,draw_vert1
	lw ra, 0(sp)
	addi sp,sp,4
	ret

# Fills the 60x80 grid with one color using successive calls to
#draw_horizontal_line
# Modifies (directly or indirectly): t0, t1, t4, a0, a1, a2, a3
draw_background:
	addi sp,sp,-4
	sw ra, 0(sp)
```

```
      li a3, BG_COLOR   #use default color
      li a1, 0    #a1= row_counter
      li t4, 60   #max rows
start0:     li a0, 0
      li a2, 79   #total number of columns
      call draw_horizontal_line  # must not modify: t4, a1, a3
      addi a1,a1, 1
      bne t4,a1, start0 #branch to draw more rows
      lw ra, 0(sp)
      addi sp,sp,4
      ret
# fills 60 x 80 gride with solid green color for successfully finding gate
draw_endground:
      addi sp,sp,-4
      sw ra, 0(sp)
      add a3, x0, s10 # color green for end game
      li a1, 0    #a1= row_counter
      li t4, 60   #max rows
start1:     li a0, 0
      li a2, 79   #total number of columns
      call draw_horizontal_line  # must not modify: t4, a1, a3
      addi a1,a1, 1
      bne t4,a1, start1 #branch to draw more rows
      lw ra, 0(sp)
      addi sp,sp,4
      ret

# draws a dot on the display at the given coordinates:
#     (X,Y) = (a0,a1) with a color stored in a3
#     (col, row) = (a0,a1)
# Modifies (directly or indirectly): t0, t1
draw_dot:
      andi t0,a0,0x7F   # select bottom 7 bits (col)
      andi t1,a1,0x3F   # select bottom 6 bits  (row)
      slli t1,t1,7      #  {a1[5:0],a0[6:0]}
      or t0,t1,t0 # 13-bit address
      sw t0, 0(s2)      # write 13 address bits to register
      sw a3, 0(s3)      # write color data to frame buffer
      ret

read_dot:
      andi t0,a0,0x7F   # select bottom 7 bits (col)
      andi t1,a1,0x3F   # select bottom 6 bits  (row)
      slli t1,t1,7      #  {a1[5:0],a0[6:0]}
      or t0,t1,t0 # 13-bit address
      sw t0, 0(s2)    # write 13 bits to register
      lw t0, 0(s11)     # load RD router from wrapper
      ret

game: addi a0, x0, 48   # set location of player block
```

```
      addi a1, x0, 58
      addi a3, x0, 0xFF # set player block color

loop: beq   t3, x0, loop    # check for interrupt flag
      lw    t4, 0(s1)       # read saved scancode
      sw    t4, 0x40(s0)    # set 7Seg
      addi  s4, s4,  1      # increment interrupt count
      sw    s4, 0x20(s0)    # output to LEDS
      # 4 possible key strokes to write dot in location and color previous
#dot as background
      bne x0, t3, BOUND     # if keystroke dected check for boundary
noact:
      csrrw x0, mie, t3     # re-enable interrupts
      addi  t3, x0, 0       # clear interrupt flag
      j     loop

 end: jal ra, draw_endground
      j end


ISR:  lw   t3, 0x100(s0)  # read scancode
      sw   t3, 0(s1)      # save to SCANCODE
      addi t3, x0, 1      # set interrupt flag
      mret

BOUND: beq t4, s5, Astroke
       beq t4, s6, Wstroke
       beq t4, s7, Sstroke
       beq t4, s8, Dstroke
       j noact

Astroke:                # move player left
      addi a0, a0, -1         # increment in direction
      jal ra, read_dot  # check for red or green block
      addi a0, a0, 1          # return to original value
      beq t0, s9, noact # if red return to loop
      beq t0, s10, end  # if green go to end and trap

      li a3, BG_COLOR   #use default color
      jal ra, draw_dot
      addi a3, x0, 0xFF # set player block color
      addi a0, a0, -1   # col
      jal ra, draw_dot
      j noact
Wstroke:          # move up
      addi a1, a1, -1   # row
      jal ra, read_dot # check for red or green block
      addi a1, a1, 1    # return to original value
      beq t0, s9, noact # if red return to loop
      beq t0, s10, end  # if green go to end and trap
```

```
        li a3, BG_COLOR   #use default color
        jal ra, draw_dot
        addi a3, x0, 0xFF # set player block color
        addi a1, a1, -1   # row
        jal ra, draw_dot
        j noact

Sstroke:          # move down
        addi a1, a1, 1    # row
        jal ra, read_dot # check for red or green block
        addi a1, a1, -1   # return to original value
        beq t0, s9, noact # if red return to loop
        beq t0, s10, end  # if green go to end and trap

        li a3, BG_COLOR   #use default color
        jal ra, draw_dot
        addi a3, x0, 0xFF # set player block color
        addi a1, a1, 1    # row
        jal ra, draw_dot
        j noact
Dstroke:          # move right
        addi a0, a0, 1    # row
        jal ra, read_dot # check for red or green block
        addi a0, a0, -1   # return to original value
        beq t0, s9, noact # if red return to loop
        beq t0, s10, end  # if green go to end and trap

        li a3, BG_COLOR   #use default color
        jal ra, draw_dot
        addi a3, x0, 0xFF # set player block color
        addi a0, a0, 1    # col
        jal ra, draw_dot
        j noact
```