BASTARD ICE CREAM

PROJECT FINAL REPORT

EMBEDDED SYSTEMS (CSEE 4840)

PROFESSOR: STEPHEN A. EDWARDS

HAODAN HUANG

DEPARTMENT OF ELECTRICAL ENGINEERING

hah2128@columbia.edu

ZIHENG ZHOU

DEPARTMENT OF ELECTRICAL ENGINEERING

zz2222@columbia.edu

LEI MAO

DEPARTMENT OF ELECTRICAL ENGINEERING

lm2833@columbia.edu

YAOZHONG SONG

DEPARTMENT OF ELECTRICAL ENGINEERING

ys2589@columbia.edu

I. INTRODUCTION

In this project, we aim to design a video game called Bastard Ice Cream. It's a one player game consists of the ice cream which controlled by the player and a fruit guard AI. The general purpose of the game is to eat up all the fruit in the map, meanwhile the player must avoid any physical touch which leads sprite dead with the fruit guard who will chases the player, the game will end if all the fruit are eaten up, or caught by the fruit guard.

II. SYSTEM ARCHITECTURE

Our project design is shown in the following block, which consists of CPU, SDRAM CONTROL, AUDIO CONTROL, VGACONTROL and PS2 CONTROL. Our project basically consists of three part of workload: VGA CONTROL, AUDIO CONTROL and software logic development.

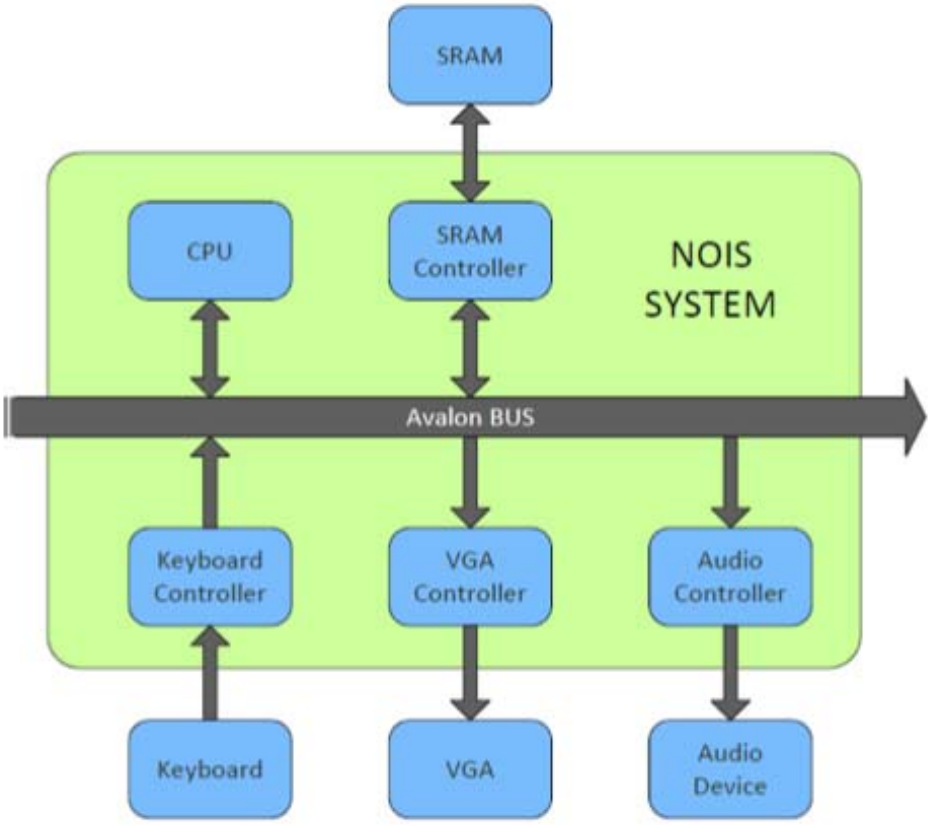

Figure 1: Implementation Architecture

III. VGA Display And Control

i) Game Image Abstraction & VGA Display

According to lab3 in the course, we noticed that we could use pictures on the Internet to feed into the FPGA and display the picture. According to the feature of the screen we are using, we divide our 640-480 pixels screen into tiles. Each 32-32 pixels construct one tile, so totally makes the screen a 20-15 tiles map.

To import characters and items of the game into our project, we use MATLAB to resize and abstract the figure data. The original figure is RGB 256*256*256 in color, but to save our memory, we use the high 5 bit of the color value which makes our color ranging from 32-256 for each.

Like VGA_R(9 downto 5) <= Sprite_R;

VGA_R(4 downto 0) <= “11111”;

This is really a high color range though.

Figure2-a: ice cream

Figure2-b: guard

To display the figure on the screen, we use three signal arrays to store the map information and according to that, our scanner (counter) on horizontal and vertical side could check whether it is inside certain tiles that need to display the corresponding items or sprites.

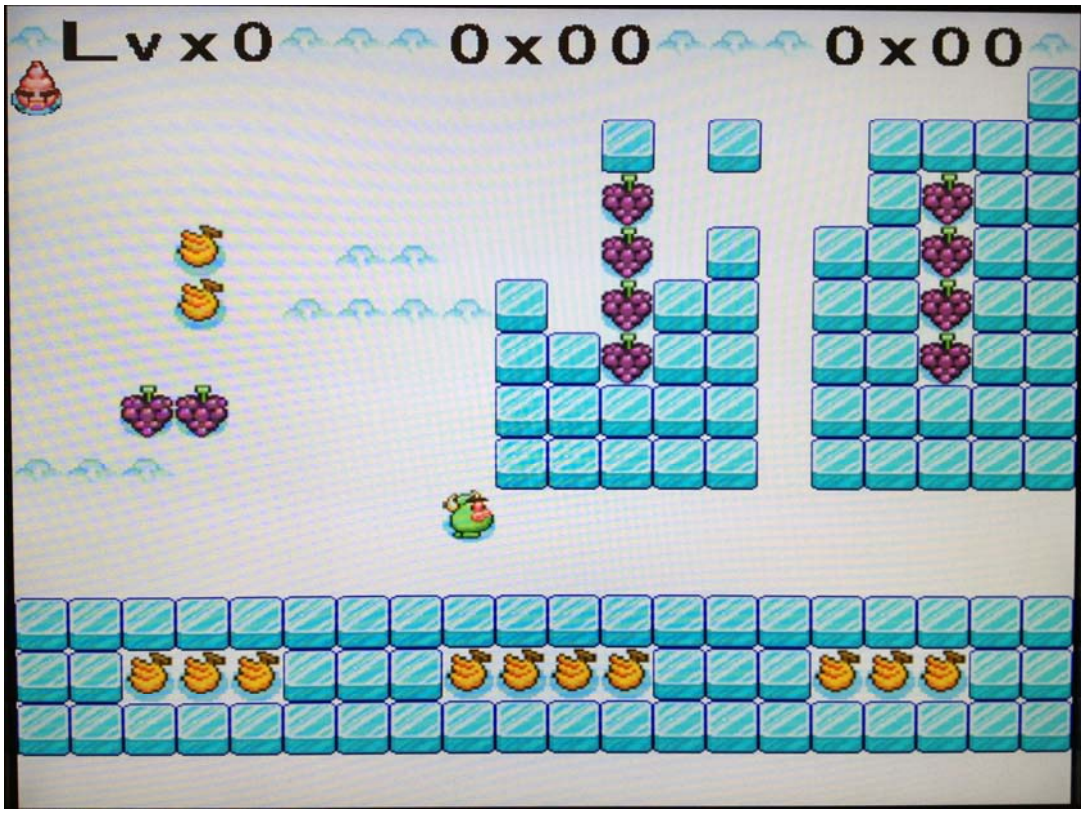

Figure3: real map

To display the under figure on the screen, we use a special number (like “00000”) to indicate that this pixel need to keep apparent for the under layer.

Difficulty & Improvement: Debugging with vhdl always consumes time, for mapping, we could make it into SDRAM or ROM to read from which will allow us to restart the game automatically. But as time limits, we didn’t achieve that point. On the other hand, I found a better solution for the picture to be shown is to change the value of the pixel on the screen when the scanner is on “BACK_PORCH” or “FRONT PORCH”, since we are not dealing with situation that the picture need to be changed all the time, this won’t affect much, but this is really a better way to do it. A pity is that we don’t really need to use such high resolution for RGB color because the original picture is really low resolution, this really affect the whole project after we finish all the picture abstraction, because our on chip memory is almost done with storing these pictures.

ii) Sprite Control & Animation

In this section, we need to use Nios-2(C) along with Quartus(vhdl) to control the select tile on the screen. In addition, keyboard need to be enabled.

To make our sprite controllable, we setup the keyboard, and use “W”,”S”,”A”,”D” keys to move the sprite ice cream “up”, ”left”, ”down”, ”right”. In software

development, we use a while loop to listen to the key information reported by hardware keyboard and react accordingly.

To introduce sprite animation shown on the screen, we need to import a large number of pictures of the sprite because every direction has three states when walking. The pictures are shown below. So in total 24 pictures are included.

Figure 4-a: motion of sprite going down

Figure 4-b: motion of guard going down

The picture are playing by hardware using vhdl to show them in order, each of them could be played for 4 pixel time. So for 32 pixels movement, the whole movement animation could be displayed for two times.

To store the data more efficient, we use ROM introduced in the slides to keep these picture data. To make the movement more real, we use pixel by pixel in the screen. But this could become a issue if we want to develop this function in software side. Unlike many games, our project has an artificial intelligence guard which need to be controlled by the program periodically. For example, player keeps pressing keyboard, in this case the guard could not move fluently.

To solve the above problem, we developed the movement of the sprite inside hardware, the software only need to give a direction, the hardware could then move pixel by pixel on the screen, after successful, it returns a signal to software. On the other hand, we set up the counter inside the hardware movement to make sure that the right state picture is selected. Finally we have this cartoon effect.

Difficulty & Improvement: This part is really tricky, because I have to separate the motion of the sprite from software to hardware. This comes up with several other problems when we are developing game logic.

IV. GAME LOGIC DEVELOPMENT

i). Sprite Skill & Level Selection

In this part, we enable the sprite with ability to generate and destroy ice tile, eat fruit. We generate three maps both in hardware and software so both hardware and software could check its own map other than transmitting a lot of data. We use unique numbers to map each kind of item on the screen. For the sake of generating and destroying ice tile, we use key “space ” to work. So once we detected an item, we check if it’s mapping value and process it accordingly.

After ice cream eats up all the fruit in one level, we will reset the position of the sprite and guard, and change the map to the next level.

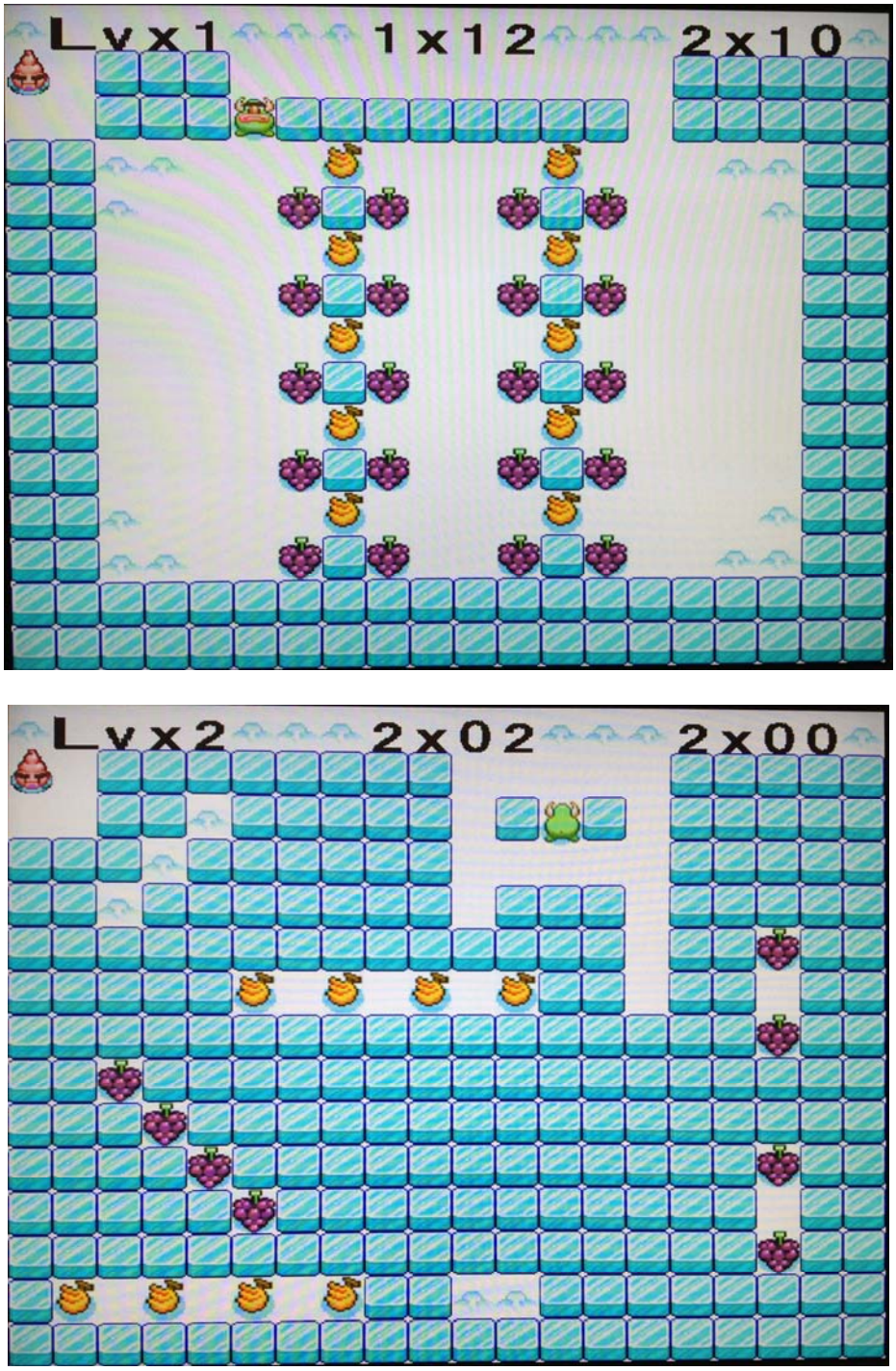

Figure 5: Level selection

If the ice cream eats up all the fruit for level 2, we will have a “WIN” character shown on the screen, otherwise a “LOSE” character will be shown.

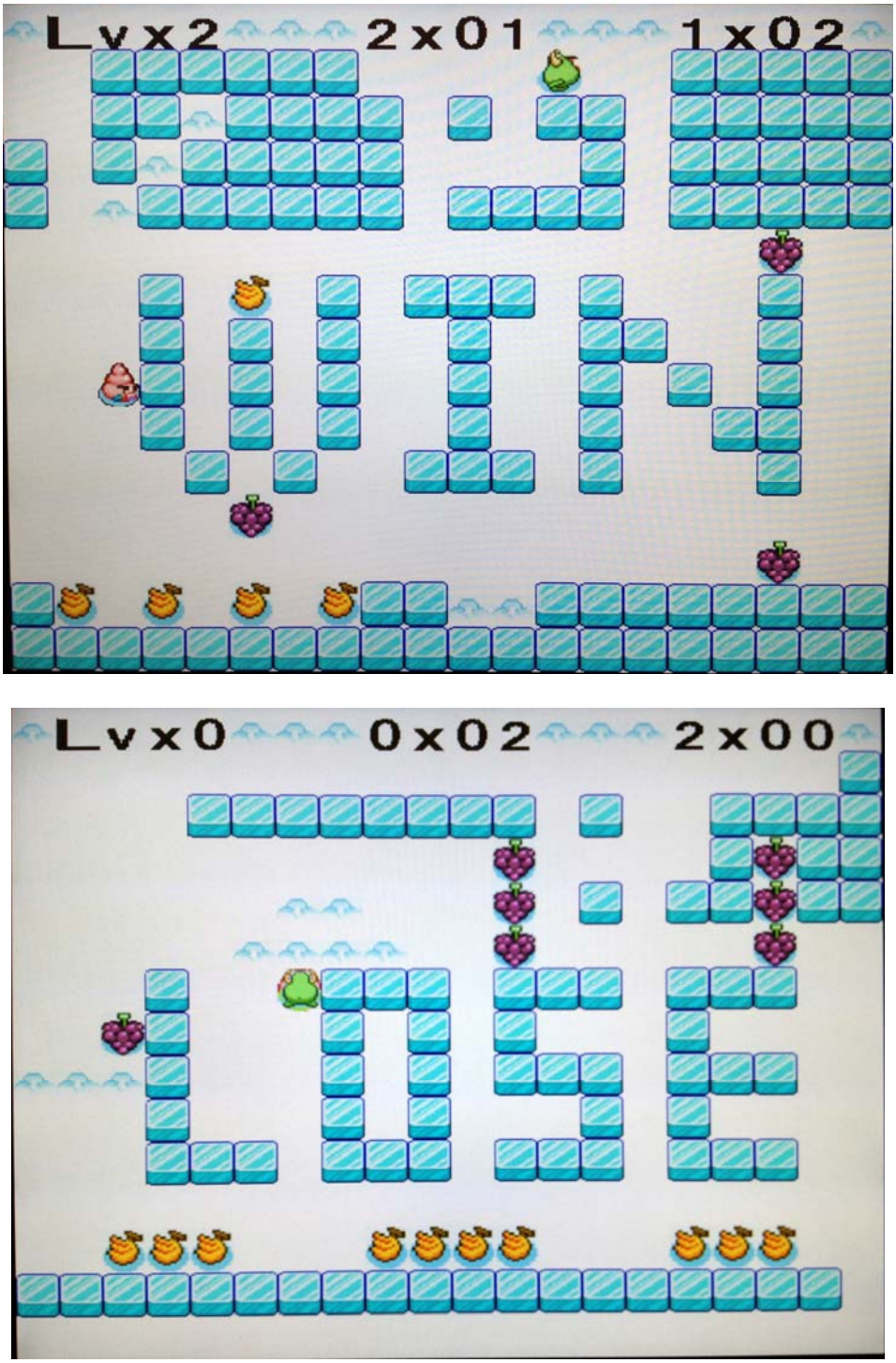

Figure 6: Win & lose Sign

Difficulty & Improvement: This part is a lot of work! The whole project depends on the logic part to integrate. When For debugging with this part, we face the problem with the actually position of the guard, the software doesn't know the exact pixel that the sprite is on, so we set up a guard lock, this ensure that the ice tile will not overlap with the guard sprite. On the other hand, if we have time, we could do the reset for the whole game, so we can start from the beginning.

ii). Artificial Intelligence Guard

The guard is implemented using interrupt. I developed two modes for guard: normal walk mode and chasing mode. I generate a random table for guard when it will walk like randomly to move around, it could also detect items that block it, the guard could eats ice, but will be blocked by fruit because it's protecting it. For the chasing mode, I developed the algorithm that the guard will chase the sprite. Moreover, it will speed up in this mode since I interacted with hardware so for animation and interrupt, both of them will become shorter which makes the guard moving very fast.

Difficulty & Improvement: The guard is the main reason that we couldn't make more fancy pictures in the game. I couldn't distribute more effort on the guard because there are many issues with logic part. To improve its performance, the guard can have more feature regarding different levels.

V. AUDIO IMPLEMENTATION

We have tried to implement our audio part in two different ways.

In our first approach, actually, we did similarly as the audio branch of lab 3. In the de2_wm8731_audio.vhd, we changed the step size of the sine wave sound, so that the frequency of the sine wave is changed, and different tones can be made. In another new higher-level file, we combined different frequencies of sound to make several sets of continuous music.

To connect with the video part of our project, we gave a number for each set of continuous music. The set of music with the number is called will be played. Theoretically, there could be 2n sets of continuous music can be produced. And for each set, there should be 2n tones combined. The factors that limit the length and diversity of this kind of music are memory and compilation time. In addition, to make it sound perfectly, we calculated the several counters in the de2_wm8731_audio.vhd, for example, "lrck", to change them to achieve rigorous frequencies, which are 262Hz, 294Hz, 330Hz, 349Hz, 392Hz, 440Hz and 493Hz. However, as the adjustment is a little complicated, we haven't made all these seven tones achieved together. This is the main reason why we didn't finally follow this approach.

Another implementation method follows this schema: we download sounds that we desire directly, instead of composing them on our own. Then we use a MATLAB to resample this wav file, by writing a script that enables generating a respective txt file containing the resampled wav represented by a sequence of hex numbers. To implement this wav sound into the VHDL code, we modified the de_wm8731_audio.vhd from lab3 to store the hex sequence in a ROM (say ROM1), which will later be assigned to sin_out, with a variable called "num" works as the controller in selecting this ROM. By the same way of implementation, we can assign

different sound effects as separate sequences of hex numbers to ROM2, ROM3, etc. The value of num is assigned from writedata, which is controlled from the nios_system via SOPC. Therefore, by setting the proper logical control, we enable sound effects when certain actions are triggered.

Difficulties and improvements: Although Lab3 has provided a way in sound creation, which we originally decided to follow, we implemented our sound effects via the second method at the end. We do have designed to generate sound effects by composing different tunes manually, but we found it very demanding in setting the precise pitch. The cost in doing this other than composing manually is that we sacrificed a little on the purity of the sound effect, i.e. some distortion exists in the final output. This actually depends a lot on the MATLAB resample rate, if we increase the rate, there will be less distortions; however, ths will result in increment on the data scale of the hex sequence, which will finally result in increment in memory usage, in considering this tradeoff, we reached an balance in setting the resample frequency to 2900Hz.

VI. CONCLUSION

In our project, we truly learned how to use vhdl which is not familiar to us at all previously. Only after we change the architecture several times do we recognize that the design and architecture is the soul of the whole project, some tiny errors or ambiguous in the design at the beginning would lead to tons of changes later. We also have a clearer vision about how to program from a hardware perspective to achieve functionality. Things need to be done in software are from instruction by instruction when only have one CPU, but hardware as state machine could really do them in parallel, they are purely different.

APPENDIX: WORK DISTRIBUTION

i) VGA Display And Control
   - Image Abstraction & VGA Display: Haodan Huang, Lei Mao, Yaozhong Song
   - Sprite Control & Animation: Haodan Huang

ii) Game Logic Development
   - Sprite Skill & Level Selection: Ziheng Zhou, Haodan Huang
   - Artificial Intelligence: Haodan Huang

iii) Audio Implementation
   - Sound abstraction and generation: Lei Mao Yaozhong Song
   - Integrate with video part: Ziheng Zhou, Yaozhong Song, Lei Mao

Source Code

lab3_vga

```
--
-- DE2 top-level module that includes the simple VGA raster generator
--
-- Stephen A. Edwards, Columbia University, sedwards@cs.columbia.edu
--
-- From an original by Terasic Technology, Inc.
-- (DE2_TOP.v, part of the DE2 system board CD supplied by Altera)
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity lab3_vga is

  port (
    -- Clocks

    CLOCK_27,                           -- 27 MHz
```

```
    CLOCK_50,                       -- 50 MHz
    EXT_CLOCK : in std_logic;       -- External Clock

    -- Buttons and switches

    KEY : in std_logic_vector(3 downto 0);    -- Push buttons
    SW : in std_logic_vector(17 downto 0);    -- DPDT switches

    -- LED displays

    HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7 -- 7-segment displays
      : out std_logic_vector(6 downto 0);
    LEDG : out std_logic_vector(8 downto 0);  -- Green LEDs
    LEDR : out std_logic_vector(17 downto 0);  -- Red LEDs

    -- RS-232 interface

    UART_TXD : out std_logic;       -- UART transmitter
    UART_RXD : in std_logic;        -- UART receiver

    -- IRDA interface

--    IRDA_TXD : out std_logic;       -- IRDA Transmitter
    IRDA_RXD : in std_logic;        -- IRDA Receiver
```

```
-- SDRAM

DRAM_DQ : inout std_logic_vector(15 downto 0); -- Data Bus
DRAM_ADDR : out std_logic_vector(11 downto 0); -- Address Bus
DRAM_LDQM,                    -- Low-byte Data Mask
DRAM_UDQM,                    -- High-byte Data Mask
DRAM_WE_N,                    -- Write Enable
DRAM_CAS_N,                   -- Column Address Strobe
DRAM_RAS_N,                   -- Row Address Strobe
DRAM_CS_N,                    -- Chip Select
DRAM_BA_0,                    -- Bank Address 0
DRAM_BA_1,                    -- Bank Address 0
DRAM_CLK,                     -- Clock
DRAM_CKE : out std_logic;          -- Clock Enable

-- FLASH

FL_DQ : inout std_logic_vector(7 downto 0);     -- Data bus
FL_ADDR : out std_logic_vector(21 downto 0);  -- Address bus
FL_WE_N,                      -- Write Enable
FL_RST_N,                     -- Reset
FL_OE_N,                      -- Output Enable
FL_CE_N : out std_logic;           -- Chip Enable
```

```
-- SRAM

SRAM_DQ : inout std_logic_vector(15 downto 0); -- Data bus 16 Bits
SRAM_ADDR : out std_logic_vector(17 downto 0); -- Address bus 18 Bits
SRAM_UB_N,                    -- High-byte Data Mask
SRAM_LB_N,                    -- Low-byte Data Mask
SRAM_WE_N,                    -- Write Enable
SRAM_CE_N,                    -- Chip Enable
SRAM_OE_N : out std_logic;            -- Output Enable

-- USB controller

OTG_DATA : inout std_logic_vector(15 downto 0); -- Data bus
OTG_ADDR : out std_logic_vector(1 downto 0);    -- Address
OTG_CS_N,                     -- Chip Select
OTG_RD_N,                     -- Write
OTG_WR_N,                     -- Read
OTG_RST_N,                    -- Reset
OTG_FSPEED,           -- USB Full Speed, 0 = Enable, Z = Disable
OTG_LSPEED : out std_logic;     -- USB Low Speed, 0 = Enable, Z = Disable
OTG_INT0,                     -- Interrupt 0
OTG_INT1,                     -- Interrupt 1
OTG_DREQ0,                    -- DMA Request 0
```

```
OTG_DREQ1 : in std_logic;                -- DMA Request 1
OTG_DACK0_N,                             -- DMA Acknowledge 0
OTG_DACK1_N : out std_logic;             -- DMA Acknowledge 1

-- 16 X 2 LCD Module

LCD_ON,                 -- Power ON/OFF
LCD_BLON,               -- Back Light ON/OFF
LCD_RW,                 -- Read/Write Select, 0 = Write, 1 = Read
LCD_EN,                 -- Enable
LCD_RS : out std_logic;     -- Command/Data Select, 0 = Command, 1 = Data
LCD_DATA : inout std_logic_vector(7 downto 0); -- Data bus 8 bits

-- SD card interface

SD_DAT,                 -- SD Card Data
SD_DAT3,                -- SD Card Data 3
SD_CMD : inout std_logic;   -- SD Card Command Signal
SD_CLK : out std_logic;     -- SD Card Clock

-- USB JTAG link

TDI,                    -- CPLD -> FPGA (data in)
TCK,                    -- CPLD -> FPGA (clk)
```

```
TCS : in std_logic;         -- CPLD -> FPGA (CS)
TDO : out std_logic;        -- FPGA -> CPLD (data out)

-- I2C bus

I2C_SDAT : inout std_logic; -- I2C Data
I2C_SCLK : out std_logic;   -- I2C Clock

-- PS/2 port

PS2_DAT,                    -- Data
PS2_CLK : in std_logic;     -- Clock

-- VGA output

VGA_CLK,                            -- Clock
VGA_HS,                             -- H_SYNC
VGA_VS,                             -- V_SYNC
VGA_BLANK,                          -- BLANK
VGA_SYNC : out std_logic;           -- SYNC
VGA_R,                              -- Red[9:0]
VGA_G,                              -- Green[9:0]
VGA_B : out std_logic_vector(9 downto 0);   -- Blue[9:0]
```

```
--  Ethernet Interface

ENET_DATA : inout std_logic_vector(15 downto 0);    -- DATA bus 16Bits
ENET_CMD,          -- Command/Data Select, 0 = Command, 1 = Data
ENET_CS_N,                              -- Chip Select
ENET_WR_N,                               -- Write
ENET_RD_N,                              -- Read
ENET_RST_N,                             -- Reset
ENET_CLK : out std_logic;                    -- Clock 25 MHz
ENET_INT : in std_logic;                    -- Interrupt

-- Audio CODEC

AUD_ADCLRCK : inout std_logic;                -- ADC LR Clock
AUD_ADCDAT : in std_logic;                   -- ADC Data
AUD_DACLRCK : inout std_logic;                -- DAC LR Clock
AUD_DACDAT : out std_logic;                  -- DAC Data
AUD_BCLK : inout std_logic;                  -- Bit-Stream Clock
AUD_XCK : out std_logic;                     -- Chip Clock

-- Video Decoder

TD_DATA : in std_logic_vector(7 downto 0);  -- Data bus 8 bits
TD_HS,                           -- H_SYNC
```

```
    TD_VS : in std_logic;                   -- V_SYNC
    TD_RESET : out std_logic;               -- Reset

    -- General-purpose I/O

    GPIO_0,                                 -- GPIO Connection 0
    GPIO_1 : inout std_logic_vector(35 downto 0) -- GPIO Connection 1
    );

end lab3_vga;

architecture datapath of lab3_vga is
signal counter : unsigned (15 downto 0);
  signal clk25 : std_logic := '0';
  signal reset_n : std_logic := '0';
  signal audio_clock : unsigned(1 downto 0) := "00";
            --signal dummy : unsigned (15 downto 0);

component de2_i2c_av_config is
  port (
    iCLK : in std_logic;
    iRST_N : in std_logic;
    I2C_SCLK : out std_logic;
    I2C_SDAT : inout std_logic
```

```
 );
 end component;

begin
  process (CLOCK_50)
 begin
   if rising_edge(CLOCK_50) then
     if counter = x"ffff" then
       reset_n <= '1';
     else
       reset_n <= '0';
       counter <= counter + 1;
     end if;
   end if;
 end process;

 process (CLOCK_50)
 begin
   if rising_edge(CLOCK_50) then
     audio_clock <= audio_clock + "1";
   end if;
 end process;

 AUD_XCK <= audio_clock(1);
```

```
--AUD_XCK <= audio_clock(1);
i2c : de2_i2c_av_config port map (
   iCLK     => CLOCK_50,
   iRST_n   => '1',
   I2C_SCLK => I2C_SCLK,
   I2C_SDAT => I2C_SDAT
  );
 nios : entity work.nios_system port map (
   clk                      => CLOCK_50,
   reset_n                  => reset_n,
   PS2_Clk_to_the_de2_ps2_0     => PS2_CLK,
   PS2_Data_to_the_de2_ps2_0    => PS2_DAT,
   SRAM_ADDR_from_the_sram      => SRAM_ADDR,
   SRAM_CE_N_from_the_sram      => SRAM_CE_N,
   SRAM_DQ_to_and_from_the_sram => SRAM_DQ,
   SRAM_LB_N_from_the_sram      => SRAM_LB_N,
   SRAM_OE_N_from_the_sram      => SRAM_OE_N,
   SRAM_UB_N_from_the_sram      => SRAM_UB_N,
   SRAM_WE_N_from_the_sram      => SRAM_WE_N,


          VGA_CLK_from_the_vga      => VGA_CLK,           -- Clock
```

```
   VGA_HS_from_the_vga      => VGA_HS,           -- H_SYNC
   VGA_VS_from_the_vga          => VGA_VS,           -- V_SYNC
   VGA_BLANK_from_the_vga    => VGA_BLANK,         -- BLANK
   VGA_SYNC_from_the_vga     => VGA_SYNC,          -- SYNC
   VGA_R_from_the_vga       => VGA_R,            -- Red[9:0]
   VGA_G_from_the_vga       => VGA_G,            -- Green[9:0]
   VGA_B_from_the_vga       => VGA_B,                                  -- Blue[9:0]



--    AUD_ADCLRCK_from_the_wm8731  => AUD_ADCLRCK,
--    AUD_ADCDAT_from_the_wm8731    => AUD_ADCDAT,
--    AUD_DACLRCK_from_the_wm8731   => AUD_DACLRCK,
--    AUD_DACDAT_from_the_wm8731   => AUD_DACDAT,
--    AUD_BCLK_from_the_wm8731     => AUD_BCLK,

   AUD_ADCDAT_to_the_wm8731 => AUD_ADCDAT,
   AUD_ADCLRCK_from_the_wm8731 => AUD_ADCLRCK,
   AUD_BCLK_to_and_from_the_wm8731  => AUD_BCLK,
   AUD_DACDAT_from_the_wm8731  => AUD_DACDAT,
   AUD_DACLRCK_from_the_wm8731 => AUD_DACLRCK

--    iCLK_from_the_de2_i2c_av_config  => CLOCK_50,
--    iRST_n_from_the_de2_i2c_av_config   => '1',
--    I2C_SCLK_from_the_de2_i2c_av_config => I2C_SCLK,
```

```
--    I2C_SDAT_from_the_de2_i2c_av_config => I2C_SDAT


   );



  HEX7    <= "1111100"; -- Leftmost

  HEX6    <= "0000110";

  HEX5    <= "1000111";

  HEX4    <= "1000111";

  HEX3    <= "1000000";

  HEX2    <= (others => '1');

  HEX1    <= (others => '1');

  HEX0    <= (others => '1');         -- Rightmost

  LEDG    <= (others => '1');

  LEDR    <= (others => '1');

  LCD_ON   <= '1';

  LCD_BLON <= '1';

  LCD_RW <= '1';

  LCD_EN <= '0';

  LCD_RS <= '0';


  SD_DAT3 <= '1';

  SD_CMD <= '1';

  SD_CLK <= '1';
```

```
UART_TXD <= '0';
DRAM_ADDR <= (others => '0');
DRAM_LDQM <= '0';
DRAM_UDQM <= '0';
DRAM_WE_N <= '1';
DRAM_CAS_N <= '1';
DRAM_RAS_N <= '1';
DRAM_CS_N <= '1';
DRAM_BA_0 <= '0';
DRAM_BA_1 <= '0';
DRAM_CLK <= '0';
DRAM_CKE <= '0';
FL_ADDR <= (others => '0');
FL_WE_N <= '1';
FL_RST_N <= '0';
FL_OE_N <= '1';
FL_CE_N <= '1';
OTG_ADDR <= (others => '0');
OTG_CS_N <= '1';
OTG_RD_N <= '1';
OTG_RD_N <= '1';
OTG_WR_N <= '1';
OTG_RST_N <= '1';
```

```
  OTG_FSPEED <= '1';
  OTG_LSPEED <= '1';
  OTG_DACK0_N <= '1';
  OTG_DACK1_N <= '1';

  TDO <= '0';

  ENET_CMD <= '0';
  ENET_CS_N <= '1';
  ENET_WR_N <= '1';
  ENET_RD_N <= '1';
  ENET_RST_N <= '1';
  ENET_CLK <= '0';

  TD_RESET <= '0';

--  I2C_SCLK <= '1';

--  AUD_DACDAT <= '1';
--  AUD_XCK <= '1';

  -- Set all bidirectional ports to tri-state
  DRAM_DQ    <= (others => 'Z');
  FL_DQ      <= (others => 'Z');
```

```
  SRAM_DQ    <= (others => 'Z');
  OTG_DATA   <= (others => 'Z');
  LCD_DATA   <= (others => 'Z');
  SD_DAT     <= 'Z';
  I2C_SDAT   <= 'Z';
  ENET_DATA  <= (others => 'Z');
  AUD_ADCLRCK <= 'Z';
  AUD_DACLRCK <= 'Z';
  AUD_BCLK   <= 'Z';
  GPIO_0     <= (others => 'Z');
  GPIO_1     <= (others => 'Z');

end datapath;
```

VGA_raster.vhd

```
--------------------------------------------------------------------------------
--
-- Simple VGA raster display
--
-- Haodan Huang & Ziheng Zhou
-- sedwards@cs.columbia.edu
--
```

```
--------------------------------------------------------------------------------

library ieee;

use ieee.std_logic_1164.all;

use ieee.numeric_std.all;


entity de2_vga_raster is


  port (

    reset_n : in std_logic;

    clk   : in std_logic;                  -- Should be 25.125 MHz


    VGA_CLK,                    -- Clock

    VGA_HS,                     -- H_SYNC

    VGA_VS,                     -- V_SYNC

    VGA_BLANK,                      -- BLANK

    VGA_SYNC : out std_logic;       -- SYNC

    VGA_R,                      -- Red[9:0]

    VGA_G,                      -- Green[9:0]

    VGA_B : out std_logic_vector(9 downto 0); -- Blue[9:0]


-------------------------------------------------------circle component

    write      : in  std_logic;

          read       : in  std_logic;

    chipselect : in  std_logic;
```

```
   address    : in  unsigned(4 downto 0);
   writedata  : in  unsigned(15 downto 0);
   readdata   : out unsigned(15 downto 0)
-------------------------------------------------------------
   );

end de2_vga_raster;

architecture rtl of de2_vga_raster is

component ice_cream_controller
 port(
clk   : in std_logic;
address: in unsigned (4 downto 0);
vertical: in integer;
horizontal: in integer;

pixel_R: out std_logic_vector(4 downto 0);
pixel_G: out std_logic_vector(4 downto 0);
pixel_B: out std_logic_vector(4 downto 0)
  );
end component;

component guard_controller
```

```
 port(
clk   : in std_logic;
address: in unsigned (4 downto 0);
vertical: in integer;
horizontal: in integer;

pixel_R: out std_logic_vector(4 downto 0);
pixel_G: out std_logic_vector(4 downto 0);
pixel_B: out std_logic_vector(4 downto 0)
  );
end component;
component info_controller
port(
clk   : in std_logic;
address: integer;
vertical: in integer;
horizontal: in integer;

pixel_R: out std_logic_vector(4 downto 0);
pixel_G: out std_logic_vector(4 downto 0);
pixel_B: out std_logic_vector(4 downto 0)
);
end component;
```

```
  -- Video parameters

  constant HTOTAL      : integer := 800;

  constant HSYNC       : integer := 96;

  constant HBACK_PORCH  : integer := 48;

  constant HACTIVE     : integer := 640;

  constant HFRONT_PORCH : integer := 16;


  constant VTOTAL      : integer := 525;

  constant VSYNC       : integer := 2;

  constant VBACK_PORCH  : integer := 33;

  constant VACTIVE     : integer := 480;

  constant VFRONT_PORCH : integer := 10;



  type ice_cream_row is array(0 to 31) of std_logic_vector(7 downto 0);   -- denote
tile content

  type ice_cream_column is array(0 to 31) of ice_cream_row;


  type snow_row is array(0 to 31) of std_logic_vector(4 downto 0);  --denote snow
content

  type snow_tile is array(0 to 31) of snow_row;


  type map_position1 is array(19 downto 0) of std_logic_vector(2 downto 0);

  type map_position is array(14 downto 0) of map_position1;
```

```
  type map_level is array (2 downto 0) of map_position;


--------------------------------------------------ice-----------------


constant ice_red : ice_cream_column :=  (

("11111101","11110110","11111101","01111000","00110000","00110111","00110111","00111000","00111000","00111000","00111000","00111000","00111000","00110111","00110111","00110111","00110111","00110111","00110111","00111000","00111000","00111000","00110111","00110111","00111000","00111000","00110111","00110100","11001010","11111101","11111111"),

("11011100","01001100","01000111","10011101","11001011","11000110","11000110","11000110","11000101","11000101","11000101","11000101","11000101","11000101","11000110","11000110","11000110","11000110","11000110","11000110","11000101","11000101","11000101","11000110","11000110","11000101","11000101","11000110","11001000","01100100","01100011","11111001"),

("01101001","10111001","10111000","11010001","11011111","11011110","11011111","11100100","10100011","01111111","10000011","10000011","10000010","10000000","11001011","11100000","11011110","11011111","11100001","11011011","10001100","10000001","01111101","10111101","11011111","10001011","01111100","10101011","11100010","11000100","10101001","01011000"),

("01001110","11011111","11100000","11011100","11011011","11100001","11000010","10000100","01110011","01101000","01100110","01100010","10010000","11000010","11010101","11011100","11100010","10110101","10000011","10000100","01100110","10000001","11000110","10011010","10000100","10111000","10111110","11001011","11011100","11100010","11000100","00110000"),

("01010001","11011011","11011100","11011101","11010111","10010110","10000001","01101000","01101001","01101111","10110010","10111000","11001011","11100000","11011111","11010000","10010001","01111011","01100100","01101100","10110100","10101111","10001100","10100111","10111010","11011010","11011111","11011110","11011101","11011111","11000010","00110101"),

("01010001","11011011","11011100","11010011","11000101","01110000","01101000","01101011","01110100","10000001","11010110","11100001","11011110","11011100","11010010","10111011","01100111","01110000","01111100","10000001","11001010","10110110","01101111","10111000","11100011","11011100","11011101","11010101","11001100","11011011","11000010","00110101"),

("01010001","11011011","11100000","10010100","01101100","01101100","01101111","01110010","10110101","11011011","11011011","11011011","11010111","110
```

10010","10000100","01101110","01110001","10011111","11010011","11001001","
01111110","10010100","11011010","11011010","11011011","11010011","1101011
0","10100101","01101010","11000110","11000101","00110101"),

("01010001","11011011","11100000","10010001","01100111","01100100","100010
11","11010110","11011001","11011101","11011111","11100100","10101100","011
01111","01100111","01111010","11011010","10101000","01101100","01110111","
11001000","11010110","11011100","11011111","11011001","01111111","0111001
1","01110000","01101001","11000101","11000101","00110101"),

("01010001","11011011","11100000","10001110","01101000","10101110","110000
11","11011101","11100000","11011011","10011000","10010010","01111100","011
00100","10101001","10110101","10001101","10100000","10110101","10110110","
11011000","11011110","11100011","10101111","10001100","01101111","0110101
0","01101011","01101011","11000101","11000101","00110101"),

("01010001","11011011","11011101","10111100","10101111","11011011","111000
00","11100011","10110011","10010100","01101000","01011111","10001001","101
10110","10011110","10011011","10100110","11000010","11100001","11011111","
11100010","11001011","10011010","01111011","01100110","01101100","0110110
0","01101100","01101011","11000101","11000101","00110101"),

("01010001","11011011","11011100","11011110","11100000","11011101","110010
00","10100010","01111011","01101001","10100000","10100110","10100110","101
00111","10100000","10101011","11011111","11011110","11011011","11011000","
10101010","10010000","01100101","01101010","01101100","01101100","0110110
0","01101100","01101011","11000101","11000101","00110101"),

("01010001","11011011","11011100","11011011","11011011","11100011","101110
00","01100011","01100110","01110000","11011001","11101011","10100110","010
11101","11000110","11100010","11011011","11011101","11100000","11011000","
01110010","01100111","01101100","01101100","01101100","01101100","0110110
0","01101100","01101011","11000101","11000101","00110101"),

("01010001","11011011","11011100","11011111","11011011","10011101","100001
01","01100010","10011100","10110111","10010011","10001110","10100000","101
10011","11010010","11011101","11100011","10111101","10010010","10010011","
01110000","01101011","01101100","01101100","01101100","01101100","0110110
0","01101001","01100101","11000100","11000101","00110101"),

("01010001","11011011","11011110","10110011","10010111","01100101","011110
11","10110001","10100011","10011101","10100100","10100011","11000001","111
00001","11100001","11010101","10100000","10000100","01100110","01101000","
01101100","01101100","01101100","01101100","01101100","01101000","0110010
0","10000101","10101100","11010011","11000011","00110101"),

("01010001","11011011","11100000","10001011","01100011","10100001","101001
11","10100110","10011111","10100000","11011010","11100001","11100000","111

00000","10110100","10011010","01101000","01101001","01101100","01101100","
01101100","01101100","01101100","01101000","01101010","10011011","1001111
1","10111111","11100111","11100001","11000001","00110101"),

("01010001","11011011","11011110","10101111","10011000","10110010","101010
01","10010000","11000110","11100011","11011011","11011101","11000110","101
01100","01110110","01100110","01101100","01101100","01101100","01101100","
01101100","01101100","01101010","10000111","10011111","11011001","1110001
1","11001001","10101011","11010100","11000011","00110101"),

("01010001","11011011","11011100","11100000","11011000","01101100","100011
00","11100100","11011101","11011011","11011100","11100001","10100100","011
00010","01101011","01101100","01101100","01101100","01101100","01101100","
01101100","01101100","01100110","10110100","11100100","11011100","1110000
1","10100111","01100000","11000100","11000101","00110101"),

("01010001","11011011","11011111","10100000","10000011","10111100","110010
11","11011100","11011110","11011000","10001101","10000101","01111001","011
01011","01101100","01101100","01101100","01101100","01101100","01101100","
01100110","10000100","11000100","11010100","11011010","10001110","1000010
1","01111000","01100110","11000100","11000101","00110101"),

("01010010","11011010","11011100","11000011","10111001","11011011","110111
11","11100010","10101001","10000111","01101100","01101000","01101010","011
01100","01101100","01101100","01101100","01101010","01100111","01101101","
10110010","11001000","11100111","10101101","10000111","01101001","0110000
1","10001100","10111110","11010110","11000010","00110110"),

("01001101","11101110","11110001","11100011","11011100","11011100","110010
11","10101110","01111101","01100000","01100111","01100111","01100111","011
00111","01100111","01100111","01100101","01111100","10010101","10011000","
11011000","11010000","10101110","01111110","01100011","10001111","1001001
0","10110101","11100000","11101111","11010010","00101011"),

("00110110","11000111","11000110","11011111","11101100","11101110","110100
00","10010110","10011100","10011101","10011101","10011101","10011101","100
11101","10011101","10011101","10011001","11000100","11110100","11110000","
11101110","11010010","10010110","10011011","10100010","11101000","1111000
1","11101110","11101011","11010011","10101011","00011000"),

("00011011","00110111","00110001","10011110","11011001","11010011","110101
00","11010111","11010111","11010111","11010111","11010111","11010111","110
10111","11010111","11010111","11010111","11010101","11010011","11010011","
11010011","11010100","11010111","11010111","11010111","11010011","1101001
1","11010011","11010101","01011100","00101101","00010110"),

("00011101","00111101","00111101","00111100","00111011","00111011","001110
11","00111011","00111011","00111011","00111011","00111011","00111011","001

```
11011","00111011","00111011","00111011","00111011","00111011","00111011","
00111011","00111011","00111011","00111011","00111011","00111011","0011101
1","00111011","00111011","00111101","00110111","00010110"),

("00011101","00111101","00111101","00111101","00111101","00111101","001111
01","00111101","00111101","00111101","00111101","00111101","00111101","001
11101","00111101","00111101","00111101","00111101","00111101","00111101","
00111101","00111101","00111101","00111101","00111101","00111101","0011110
1","00111101","00111101","00111110","00110111","00010110"),

("00011101","00111101","00111101","00111101","00111101","00111101","001111
01","00111101","00111101","00111101","00111101","00111101","00111101","001
11101","00111101","00111101","00111101","00111101","00111101","00111101","
00111101","00111101","00111101","00111101","00111101","00111101","0011110
1","00111101","00111101","00111110","00110111","00010110"),

("00011101","00111101","00111101","00111101","00111101","00111101","001111
01","00111101","00111101","00111101","00111101","00111101","00111101","001
11101","00111101","00111101","00111101","00111101","00111101","00111101","
00111101","00111101","00111101","00111101","00111101","00111101","0011110
1","00111101","00111101","00111110","00110111","00010110"),

("00011101","00111101","00111101","00111101","00111101","00111101","001111
01","00111101","00111101","00111101","00111101","00111101","00111101","001
11101","00111101","00111101","00111101","00111101","00111101","00111101","
00111101","00111101","00111101","00111101","00111101","00111101","0011110
1","00111101","00111101","00111110","00110111","00010110"),

("00011101","00111101","00111101","00111101","00111101","00111101","001111
01","00111101","00111101","00111101","00111101","00111101","00111101","001
11101","00111101","00111101","00111101","00111101","00111101","00111101","
00111101","00111101","00111101","00111101","00111101","00111101","0011110
1","00111101","00111101","00111110","00110111","00010110"),

("00011000","00111110","00111110","00111110","00111110","00111110","001111
10","00111110","00111110","00111110","00111110","00111110","00111110","001
11110","00111110","00111110","00111110","00111110","00111110","00111110","
00111110","00111110","00111110","00111110","00111110","00111110","0011111
0","00111110","00111110","00111111","00110111","00010000"),

("00111101","00110010","00110010","00110111","00111001","00111001","001110
01","00111001","00111001","00111001","00111001","00111001","00111001","001
11001","00111001","00111001","00111001","00111001","00111001","00111001","
00111001","00111001","00111001","00111001","00111001","00111001","0011100
1","00111001","00111001","00110011","00110100","00111111"),

("11010111","00110011","00110010","00100111","00100001","00100010","001000
10","00100010","00100010","00100010","00100010","00100010","00100010","001
```

00010","00100010","00100010","00100010","00100010","00100010","00100010","
00100010","00100010","00100010","00100010","00100010","00100010","0010001
0","00100010","00100010","00101010","01010001","11111001"),

("11111101","11110110","11111110","01100010","00001101","00010101","000101
01","00010101","00010101","00010101","00010101","00010101","00010101","000
10101","00010101","00010101","00010101","00010101","00010101","00010101","
00010101","00010101","00010101","00010101","00010101","00010101","0001010
1","00010100","00010010","11000010","11111110","11111111")

);

constant ice_green : ice_cream_column := (

("11111101","11110110","11111101","01110000","00100011","00101011","001010
11","00101011","00101011","00101011","00101011","00101011","00101011","001
01011","00101011","00101011","00101011","00101011","00101011","00101011","
00101011","00101011","00101011","00101011","00101011","00101011","0010101
1","00101011","00101000","11000111","11111101","11111111"),

("11011001","01000001","00111001","10101000","11100100","11011110","110111
10","11011110","11011110","11011110","11011110","11011110","11011110","110
11110","11011110","11011110","11011110","11011110","11011110","11011110","
11011110","11011110","11011110","11011110","11011110","11011110","1101111
0","11011110","11100000","01100000","01011000","11111001"),

("01100011","11001110","11001101","11101100","11111110","11111100","111111
00","11111101","11110011","11101101","11101110","11101110","11101110","111
01110","11111001","11111100","11111100","11111100","11111100","11111011","
11101111","11101110","11101101","11110111","11111100","11101111","1110110
1","11110100","11111101","11011011","10111001","01001101"),

("01001000","11111100","11111110","11111010","11111000","11111001","111101
00","11101011","11101000","11100111","11100110","11100101","11101101","111
10101","11110111","11111000","11111010","11110010","11101010","11101011","
11100110","11101010","11110101","11101110","11101011","11110011","1111010
0","11110110","11111000","11111111","11011100","00100011"),

("01001011","11110111","11111001","11111000","11110111","11101101","111010
10","11100111","11100111","11101000","11110010","11110011","11110101","111
11000","11111000","11110110","11101100","11101001","11100110","11100111","
11110010","11110001","11101100","11110000","11110011","11110111","1111100
0","11111000","11111000","11111100","11011000","00101000"),

("01001011","11110111","11111001","11110111","11110101","11100111","11100110","11100111","11101000","11101010","11111000","11111001","11111001","11111000","11110111","11110011","11100110","11100111","11101001","11101010","11110101","11110010","11101000","11110011","11111010","11111000","11111000","11110111","11110110","11111100","11011000","00101000"),

("01001011","11110111","11111001","11101101","11100111","11100111","11100111","11100111","11110010","11111000","11111000","11111000","11111000","11110111","11101011","11101000","11100111","11101111","11110111","11110101","11101010","11101101","11110111","11111000","11111000","11110111","11111000","11110000","11100111","11111001","11011001","00101000"),

("01001011","11110111","11111010","11101101","11100110","11100101","11101011","11111000","11111000","11111000","11111001","11111010","11110001","11100111","11100110","11101001","11110111","11110000","11100111","11101001","11110110","11111000","11111000","11111001","11111000","11101010","11101000","11101000","11100111","11111001","11011001","00101000"),

("01001011","11110111","11111010","11101100","11100110","11110001","11110100","11111000","11111001","11111000","11101110","11101101","11101010","11100110","11110000","11110010","11101100","11101111","11110010","11110010","11110111","11111000","11111010","11110010","11101100","11101000","11100111","11100111","11111001","11011001","00101000"),

("01001011","11110111","11111001","11110011","11110001","11111000","11111001","11111001","11110010","11101101","11100111","11100101","11101100","11110011","11101111","11101110","11110000","11110100","11111001","11111001","11111001","11110110","11101110","11101001","11100110","11100111","11100111","11100111","11111001","11011001","00101000"),

("01001011","11110111","11111001","11111001","11111001","11111000","11110101","11101111","11101001","11100110","11101111","11110000","11110000","11110000","11101111","11110001","11111001","11111001","11111000","11110111","11110000","11101100","11100110","11100111","11100111","11100111","11100111","11100111","11111001","11011001","00101000"),

("01001011","11110111","11111001","11111000","11111000","11111010","11110011","11100101","11100110","11100111","11110111","11111010","11110000","11100101","11110101","11111001","11111000","11111000","11111001","11111000","11101000","11100110","11100111","11100111","11100111","11100111","11100111","11100111","11111001","11011001","00101000"),

("01001011","11110111","11111001","11111001","11111000","11101110","11101011","11100101","11101110","11110010","11101101","11101101","11101111","11110010","11110111","11111000","11111010","11110011","11101101","11101101","11101000","11100111","11100111","11100111","11100111","11100111","11100111","11100111","11100110","11111000","11011001","00101000"),

("01001011","11110111","11111001","11110010","11101110","11100110","11101001","11110001","11101111","11101110","11110000","11110000","11110100","11111001","11111001","11110111","11110000","11101011","11100110","11100110","11100111","11100111","11100111","11100111","11100111","11100110","11100110","11101010","11110000","11111011","11011000","00101000"),

("01001011","11110111","11111010","11101100","11100110","11101111","11110000","11110000","11101111","11101111","11111000","11111001","11111001","11111001","11110010","11101110","11100110","11100110","11100111","11100111","11100111","11100111","11100110","11100110","11101110","11101111","11110100","11111001","11111101","11011000","00101000"),

("01001011","11110111","11111001","11110001","11101110","11110010","11110000","11101100","11110101","11111001","11111000","11111000","11110101","11110001","11101000","11100110","11100111","11100111","11100111","11100111","11100111","11100111","11100111","11101011","11101111","11111000","11111010","11110101","11110001","11111011","11011000","00101000"),

("01001011","11110111","11111001","11111001","11111000","11100111","11101100","11111010","11111000","11111000","11111000","11111001","11101111","11100101","11100111","11100111","11100111","11100111","11100111","11100111","11100111","11100111","11100110","11110010","11111010","11111000","11111001","11110000","11100101","11111000","11011001","00101000"),

("01001011","11110111","11111001","11101111","11101010","11110011","11110101","11111000","11111000","11110111","11101100","11101011","11101001","11100111","11100111","11100111","11100111","11100111","11100111","11100111","11100111","11101011","11110100","11110111","11110111","11101100","11101011","11101001","11100111","11111001","11011001","00101000"),

("01001100","11110111","11111001","11110101","11110011","11111000","11111000","11111010","11110001","11101011","11100111","11100111","11100111","11100111","11100111","11100111","11100111","11100110","11100111","11110010","11110101","11111010","11110001","11101011","11100111","11100101","11101100","11110011","11111011","11011001","00101001"),

("01000110","11111100","11111110","11111010","11111000","11111000","11110110","11110001","11101010","11100110","11100110","11100110","11100110","11100110","11100110","11100110","11101001","11101101","11101110","11110111","11110110","11110001","11101010","11100110","11101100","11101101","11110010","11111000","11111111","11011100","00100001"),

("00111000","11101001","11101010","11110110","11111101","11111100","11111000","11101111","11110000","11110000","11110000","11110000","11110000","11110000","11110000","11110000","11110000","11101111","11110110","11111110","11111101","11111101","11111000","11101111","11101111","11110001","11111100","11111101","11111101","11111100","11110011","11001001","00010100"),

("00101111","10110111","10110110","11011100","11110001","11101111","111011
11","11110000","11110000","11110000","11110000","11110000","11110000","111
10000","11110000","11110000","11110000","11110000","11101111","11101111","
11101111","11101111","11110000","11110000","11110000","11101111","1110111
1","11101111","11110000","11000111","10011100","00010100"),

("00110000","10111001","10111010","10111000","10110111","10110111","101101
11","10111000","10110111","10110111","10110111","10110111","10110111","101
11000","10111000","10111000","10110111","10110111","10111000","10111000","
10111000","10111000","10111000","10110111","10110111","10110111","1011011
1","10110111","10110111","10111100","10100000","00010100"),

("00110000","10111001","10111001","11000010","11000111","11001000","110000
11","10110111","11000001","11000111","11001000","11001001","11000001","101
11000","10111000","10111011","11001000","11000001","10111000","10111001","
10111001","10111001","10110111","11000001","11000111","11000110","1100011
0","11000110","11000110","10111111","10011111","00010100"),

("00110000","10111001","10111000","11010101","11100100","11011001","110100
01","11000010","11011000","11100011","11011001","11011001","11001001","101
10111","11000010","11001001","11011001","11001010","10111000","10111001","
10111000","10111100","11000011","11010111","11100011","11100010","1110001
0","11100011","11100100","11000110","10011111","00010100"),

("00110000","10111001","10111001","11001110","11010111","11000001","110010
11","11100101","11011101","11010110","10111010","10110110","10111100","110
00010","11010110","11010101","10111000","10111000","10111000","10111001","
10111111","11001011","11100011","11100001","11100001","11100010","1110001
0","11011110","11011001","11000100","10011111","00010100"),

("00110000","10111001","10111010","10111101","11000010","11011110","111000
00","11011100","11000101","10111000","10111110","10111110","11001100","110
11100","11000000","10111000","10111001","10111100","10111111","11000001","
11011110","11100010","11100001","11100001","11100001","11011100","1101110
0","11001100","10111000","10111101","10100000","00010100"),

("00110000","10111001","10111000","11010011","11100010","11100100","110101
10","10111001","10111000","10111010","11011110","11100100","11001111","101
11000","10111001","10111001","10110101","11001010","11100010","11100000","
11100001","11100001","11100001","11100011","11100001","10111110","1011100
1","10111010","10111001","10111101","10100000","00010100"),

("00101100","10111101","10111100","11010110","11100010","11001111","110001
10","10111000","11000111","11001111","11001110","11001111","11000100","101
11000","10111001","10111101","11001101","11010111","11100011","11100010","
11100010","11100010","11100011","11010101","11001011","10111011","1011100
1","10111001","10111001","11000000","10100010","00001110"),

("01001100","10010111","10010101","11000100","11011011","10111100","101110
01","10111001","11010001","11011100","10111100","10111000","10111001","101
11010","10111001","11000000","11011101","11011100","11011011","11011011","
11011011","11011011","11011100","11000110","10111000","10111010","1011101
0","10111010","10111010","10100001","10001000","00111100"),

("11010110","00110000","00101010","01111000","10100001","10011100","100111
00","10011100","10011101","10011101","10011100","10011100","10011100","100
11100","10011100","10011100","10011101","10011101","10011101","10011101","
10011101","10011101","10011101","10011100","10011100","10011100","1001110
0","10011100","10011101","01000100","01001011","11111001"),

("11111101","11110101","11111101","01100010","00001110","00010110","000101
10","00010110","00010110","00010110","00010110","00010110","00010110","000
10110","00010110","00010110","00010110","00010110","00010110","00010110","
00010110","00010110","00010110","00010110","00010110","00010110","0001011
0","00010101","00010011","11000001","11111101","11111111")

);

constant ice_blue : ice_cream_column := (

("11111110","11111100","11111110","11010110","11000000","11000010","110000
10","11000010","11000010","11000010","11000010","11000010","11000010","110
00010","11000010","11000010","11000010","11000010","11000010","11000010","
11000010","11000010","11000010","11000010","11000010","11000010","1100001
0","11000010","11000001","11101111","11111110","11111111"),

("11110101","11001000","11000110","11100011","11110011","11110001","111100
01","11110001","11110001","11110001","11110001","11110001","11110001","111
10001","11110001","11110001","11110001","11110001","11110001","11110001","
11110001","11110001","11110001","11110001","11110001","11110001","1111000
1","11110001","11110010","11010000","11001111","11111110"),

("11010001","11101101","11101101","11110101","11111001","11111001","111110
01","11111001","11110001","11101101","11101101","11101101","11101101","111
01101","11110110","11111001","11111001","11111001","11111001","11111000","
11101110","11101101","11101101","11110101","11111001","11101110","1110110
0","11110010","11111010","11110000","11101000","11001100"),

("11001010","11111001","11111001","11111000","11111000","11111000","111101
00","11101101","11101011","11101010","11101001","11101001","11101110","111
10100","11110111","11111000","11111000","11110011","11101101","11101101","

11101001","11101101","11110110","11110000","11101101","11110011","1111010
0","11110110","11111000","11111010","11110000","11000000"),

("11001011","11111000","11111000","11111000","11110111","11101111","111011
01","11101010","11101010","11101011","11110011","11110100","11110110","111
11000","11111000","11110110","11101110","11101100","11101001","11101010","
11110011","11110010","11101110","11110001","11110011","11110111","1111100
0","11111000","11111000","11111001","11110000","11000001"),

("11001011","11111000","11111000","11110111","11110101","11101011","111010
10","11101010","11101011","11101101","11110111","11111000","11111000","111
11000","11110111","11110100","11101001","11101010","11101100","11101101","
11110101","11110011","11101010","11110011","11111001","11111000","1111100
0","11110111","11110110","11111001","11110000","11000001"),

("11001011","11111000","11111001","11101111","11101001","11101010","111010
10","11101011","11110011","11111000","11111000","11111000","11111000","111
10111","11101101","11101010","11101011","11110000","11110111","11110101","
11101101","11101111","11111000","11111000","11111000","11110111","1111100
0","11110001","11101001","11110110","11110000","11000001"),

("11001011","11111000","11111001","11101111","11101001","11101001","111011
10","11110111","11111000","11111000","11111001","11111010","11110010","111
01010","11101001","11101011","11110111","11110001","11101010","11101100","
11110110","11110111","11111000","11111001","11111000","11101100","1110101
1","11101011","11101010","11110110","11110000","11000001"),

("11001011","11111000","11111001","11101110","11101001","11110010","111101
01","11111000","11111001","11111000","11110000","11101111","11101100","111
01001","11110010","11110011","11101110","11110000","11110011","11110011","
11110111","11111000","11111001","11110011","11101110","11101011","1110101
0","11101010","11101010","11110110","11110000","11000001"),

("11001011","11111000","11111000","11110100","11110010","11111000","111110
01","11111001","11110011","11101111","11101010","11101000","11101110","111
10011","11110001","11110000","11110010","11110101","11111001","11111001","
11111001","11110110","11110000","11101100","11101001","11101010","1110101
0","11101010","11101010","11110110","11110000","11000001"),

("11001011","11111000","11111000","11111001","11111001","11111000","111101
10","11110001","11101100","11101001","11110000","11110001","11110001","111
10001","11110001","11110010","11111001","11111001","11111000","11111000","
11110010","11101111","11101001","11101010","11101010","11101010","1110101
0","11101010","11101010","11110110","11110000","11000001"),

("11001011","11111000","11111000","11111000","11111000","11111001","111101
00","11101000","11101001","11101010","11111000","11111010","11110001","111
01000","11110110","11111001","11111000","11111000","11111001","11111000","

11101011","11101001","11101010","11101010","11101010","11101010","1110101
0","11101010","11101010","11110110","11110000","11000001"),

("11001011","11111000","11111000","11111001","11111000","11110000","111011
01","11101000","11110000","11110011","11101111","11101111","11110001","111
10011","11110111","11111000","11111001","11110100","11101111","11101111","
11101010","11101010","11101010","11101010","11101010","11101010","1110101
0","11101010","11101001","11110110","11110000","11000001"),

("11001011","11111000","11111001","11110011","11101111","11101001","111011
00","11110010","11110001","11110000","11110001","11110001","11110101","111
11001","11111001","11110111","11110000","11101101","11101001","11101001","
11101010","11101010","11101010","11101010","11101010","11101001","1110100
1","11101101","11110010","11111000","11110000","11000001"),

("11001011","11111000","11111001","11101110","11101001","11110000","111100
01","11110001","11110000","11110001","11111000","11111001","11111001","111
11001","11110011","11101111","11101001","11101001","11101010","11101010","
11101010","11101010","11101010","11101001","11101001","11110000","1111000
1","11110101","11111001","11111001","11110000","11000001"),

("11001011","11111000","11111001","11110011","11110000","11110011","111100
10","11101110","11110110","11111001","11111000","11111000","11110101","111
10010","11101011","11101001","11101010","11101010","11101010","11101010","
11101010","11101010","11101010","11101110","11110001","11111000","1111100
1","11110110","11110010","11111000","11110000","11000001"),

("11001011","11111000","11111000","11111001","11111000","11101010","111011
10","11111010","11111000","11111000","11111000","11111001","11110001","111
01000","11101010","11101010","11101010","11101010","11101010","11101010","
11101010","11101010","11101001","11110011","11111010","11111000","1111100
1","11110001","11101000","11110110","11110000","11000001"),

("11001011","11111000","11111001","11110001","11101101","11110100","111101
10","11111000","11111000","11110111","11101110","11101101","11101100","111
01010","11101010","11101010","11101010","11101010","11101010","11101010","
11101010","11101101","11110101","11110111","11111000","11101110","1110110
1","11101100","11101010","11110110","11110000","11000001"),

("11001110","11111000","11111000","11110101","11110100","11111000","111110
00","11111001","11110010","11101101","11101010","11101010","11101010","111
01010","11101010","11101010","11101010","11101010","11101001","11101010","
11110011","11110110","11111010","11110010","11101101","11101010","1110100
0","11101110","11110100","11111000","11110000","11000101"),

("10101011","11111101","11111110","11111010","11111000","11111000","111101
10","11110010","11101100","11101001","11101001","11101001","11101001","111
01001","11101001","11101001","11101001","11101100","11101111","11110000","

11110111","11110110","11110010","11101100","11101001","11101110","1110111
1","11110011","11111000","11111110","11101110","10011010"),

("01101101","11101110","11101110","11110111","11111101","11111100","111110
00","11110001","11110010","11110010","11110010","11110010","11110010","111
10010","11110010","11110010","11110010","11110111","11111100","11111100","
11111100","11111001","11110001","11110010","11110011","11111011","1111110
0","11111100","11111100","11110101","11010110","01010010"),

("01101000","11000100","11000011","11100010","11110100","11110010","111100
10","11110010","11110010","11110010","11110010","11110010","11110010","111
10010","11110010","11110010","11110010","11110010","11110010","11110010","
11110010","11110010","11110010","11110010","11110010","11110010","1111001
0","11110010","11110011","11010001","10110010","01010101"),

("01101001","11000110","11000111","11000101","11000100","11000100","110001
00","11000101","11000100","11000100","11000100","11000100","11000100","110
00101","11000101","11000101","11000100","11000100","11000101","11000101","
11000101","11000101","11000101","11000100","11000100","11000100","1100010
0","11000100","11000100","11001000","10110101","01010101"),

("01101001","11000110","11000110","11001111","11010100","11010100","110011
11","11000100","11001110","11010100","11010100","11010101","11001101","110
00101","11000101","11000111","11010100","11001110","11000101","11000110","
11000110","11000110","11000100","11001110","11010100","11010011","1101001
1","11010011","11010011","11001011","10110100","01010101"),

("01101001","11000110","11000101","11100001","11101111","11100100","110111
00","11001110","11100100","11101110","11100100","11100100","11010101","110
00100","11001110","11010100","11100100","11010101","11000101","11000110","
11000101","11001000","11001111","11100010","11101110","11101101","1110110
1","11101110","11101111","11010010","10110100","01010101"),

("01101001","11000110","11000101","11011001","11100011","11001110","110101
11","11110000","11101000","11100010","11000111","11000011","11001001","110
01111","11100001","11100000","11000101","11000101","11000101","11000110","
11001100","11010111","11101110","11101100","11101100","11101101","1110110
1","11101001","11100100","11001111","10110100","01010101"),

("01101001","11000110","11000110","11001001","11001110","11101010","111010
10","11100111","11010001","11000101","11001011","11001011","11011001","111
01000","11001101","11000101","11000110","11001001","11001100","11001110","
11101001","11101100","11101100","11101100","11101100","11100111","1110011
1","11011000","11000101","11001000","10110101","01010101"),

("01101001","11000110","11000101","11011110","11101101","11101111","111000
01","11000110","11000101","11000111","11101001","11101111","11011011","110
00101","11000110","11000110","11000010","11010110","11101101","11101011","

11101100","11101100","11101100","11101110","11101100","11001011","1100011
0","11000111","11000110","11001000","10110101","01010101"),

("01100110","11001001","11001000","11100001","11101101","11011011","110100
11","11000101","11010011","11011011","11011001","11011010","11010000","110
00101","11000110","11001001","11011001","11100011","11101110","11101101","
11101101","11101101","11101110","11100001","11010111","11001000","1100011
0","11000110","11000110","11001011","10110111","01010010"),

("01111101","10101111","10101101","11010100","11100111","11001001","110001
10","11000111","11011110","11101000","11001001","11000101","11000111","110
01000","11000111","11001110","11101001","11101000","11100111","11100111","
11100111","11100111","11101000","11010010","11000101","11001000","1100100
0","11001000","11001000","10110110","10100101","01110011"),

("11100010","01101001","01100110","10011100","10111010","10110111","101101
11","10110111","10110111","10110111","10110111","10110111","10110111","101
10111","10110111","10110111","10110111","10110111","10110111","10110111","
10110111","10110111","10110111","10110111","10110111","10110111","1011011
1","10110111","10111000","01111000","01111101","11111011"),

("11111110","11111000","11111110","10001110","01010001","01010111","010101
11","01010111","01010111","01010111","01010111","01010111","01010111","010
10111","01010111","01010111","01010111","01010111","01010111","01010111","
01010111","01010111","01010111","01010111","01010111","01010111","0101011
1","01010111","01010101","11010011","11111110","11111111")

);

----------banana

constant banana_red : ice_cream_column := (

("11110110","11110110","11110110","11110110","11110110","11110110","111101
10","11110110","11110110","11110110","11110110","11110110","11110110","111
10110","11110111","11110010","11001011","11000110","11000111","11000111","
11000111","11001001","11110010","11111000","11110111","11110110","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","111101
10","11110110","11110110","11110110","11110110","11110110","11110110","111
10110","11111000","11101001","01010111","01001000","01001010","01001010","
01001000","01010010","11100010","11111001","11110110","11110110","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","11110111","11111100","11111100","11111100","11110111","11110110","11110110","11110110","11101000","01010010","01011011","10010000","10011100","10100110","10100010","01110011","01101011","01100110","10111011","11111111","11111100","11111100","11111001","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11111001","11101001","10110111","10111000","10111000","11101010","11111001","11110111","11110111","11111000","11101110","10010100","10000111","10000000","10011010","10101100","10100111","01111000","01110000","01101110","10011000","10111011","10111001","10110111","11010011","11111001","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","11001100","01001011","01001100","01001101","11010000","11110101","11110001","11110001","11110100","11110111","11111111","11000110","01000011","01101111","10001000","10001000","10110100","10111011","11000001","01111110","01000101","01001000","01000001","10010010","11111101","11110110"),

("11110110","11110110","11110110","11110111","11110001","01110101","01111101","11011010","11100001","11101000","01111110","01100001","01100100","01100000","11001001","11111011","11111100","11000010","01000111","01110001","10001011","10000110","01011111","01101000","10010000","10010101","10011001","01011011","01000101","10010101","11111101","11110110"),

("11110110","11110110","11110110","11110111","11110001","01011101","01101110","11110011","11111010","11111111","01101011","01000001","01000110","01000001","11000011","11111111","11111111","11000100","01000010","01101111","10001010","10000100","01010001","01011010","10001001","10011011","10101010","01011110","01000101","10010101","11111101","11110110"),

("11110110","11110110","11110110","11110111","11110001","01011100","01101100","11110011","11110011","11110100","11010110","11001101","11001110","11010001","10010010","01110100","01110100","10010010","11010011","10101011","10010011","10010010","01010110","01010000","01011001","01011110","01100001","01001101","01000001","10010010","11111101","11110110"),

("11110110","11110110","11110110","11110111","11110010","10001100","10001010","10111101","11010110","11101111","11111011","11111111","11111111","11111111","10100011","01110101","01110101","10100011","11111111","11010101","10111000","10110101","01011000","01010111","01111101","01111011","01111010","01111100","01111000","10110000","11111011","11110110"),

("11110110","11110110","11110110","11110110","11110110","11111110","11010010","01000001","10011100","11110001","11101110","11101111","11101111","11101111","11111010","11111111","11111111","11111111","11111111","11111111","11111111","11110111","01011101","01110110","11111101","11111010","11111010","11111010","11111000","11110110","11110110"),

("11110110","11110110","11110110","11110111","11110010","01111111","100001
10","11011100","11101100","11111100","11101101","11101010","11101010","111
01010","11110000","11110011","11110011","11110001","11101100","11111000","
11111111","11111101","11100010","10111111","01101000","10111011","1111111
1","11110111","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110111","11111000","11110010","01011100","011100
00","11111111","11111111","11111111","11110101","11110010","11110010","111
10010","11101111","11101101","11101101","11101110","11110001","11111001","
11111111","11111111","11111111","11010001","00111111","10010011","1101100
0","11110000","11111000","11110111","11110110","11110110"),

("11110110","11110110","11110100","11110100","11101110","01100000","011011
10","11101010","11110100","11111111","11111111","11111111","11111111","111
11111","11110000","11101001","11101001","11110000","11111111","11111110","
11111110","11111110","11111111","11010100","01001101","01110001","1000111
0","11100000","11110101","11110100","11110110","11110110"),

("11110110","11111010","11000100","10011100","10011100","01011000","010100
01","01011101","10101100","11110111","11110001","11110011","11111111","111
11111","11111101","11111100","11111100","11111101","11111111","11110001","
11100111","11101001","11111101","11111000","11100110","10001101","0100001
1","10010000","10011101","11000100","11111010","11110110"),

("11110110","11111010","10111100","10001101","10001111","01010100","010010
00","01000000","10011110","11110110","11101111","11110001","11111111","111
11111","11111111","11111111","11111111","11111111","11111111","11101111","
11100011","11100101","11111100","11111110","11111111","10010010","0011010
1","10000000","10001101","10111011","11111010","11110110"),

("11110111","11111111","10011010","01001101","01011001","10111110","110100
00","11001101","11011101","11101100","11101011","11101100","11110100","111
10100","11110100","11110011","11101011","11101010","11101001","11110010","
11111000","11111000","11111110","11111101","11111111","10010010","0011011
1","10000010","10001111","10111101","11111101","11110111"),

("11101110","11100000","01111110","00110011","01000001","11100101","111111
11","11111111","11110111","11101111","11110000","11110000","11101101","111
01101","11101101","11101101","11101010","11101010","11101001","11110111","
11111111","11111111","11111101","11111011","11111101","10010001","0011011
1","10000010","10010000","10110000","11011100","11101110"),

("11011001","10001111","01100000","00111001","01000100","11010111","111101
10","11111111","11111111","11111111","11111111","11111100","11101010","111
01010","11101010","11101100","11111101","11111111","11111111","11111111","
11111111","11111111","11111000","11110101","11110110","10001110","0011011
1","10000010","10010011","10010000","10001011","11011001"),

("11011010","10001111","10000111","10000000","01111111","01100010","10000010","11110011","11111001","11111111","11111111","11111110","11111011","11111011","11111011","11111011","11111111","11111111","11111111","11111111","11111110","11110111","11001100","01011101","01110010","10000001","10001110","10010001","10010001","10001110","11011010"),

("11011010","10001110","10010010","10010100","10010000","01000000","01100000","11101101","11110011","11111010","11111011","11111101","11111111","11111111","11111111","11111111","11111111","11111110","11111100","11111010","11111001","11111000","11110100","11000001","00111000","01101101","10010111","10010001","10010001","10010001","10001110","11011010"),

("11011010","10001110","10010001","10010010","10001110","01000010","01100011","11101111","11101110","11101111","11110101","11111000","11111111","11111111","11111111","11111111","11111111","11111101","11110111","11110010","11101111","11101111","11110010","11000101","01000111","01110010","10010100","10010001","10010001","10010001","10001110","11011010"),

("11011010","10001110","10010001","10010001","10001111","01010100","01101010","11001010","11011101","11110000","11110100","11110111","11111100","11111100","11111101","11111101","11111100","11111010","11110110","11110001","11101110","11101100","11001110","10101011","01011000","01111001","10010011","10010001","10010001","10010001","10001110","11011010"),

("11011001","10001011","10010000","10010001","10010001","10010011","10000001","01000100","10011111","11110100","11110010","11110011","11101111","11101111","11110101","11110111","11101111","11110000","11110011","11101111","11101110","11100101","01001110","01001110","10010011","10010001","10010001","10010001","10010000","10001011","11011001"),

("11101110","11011100","10110000","10001110","10010001","10010010","10001101","01111111","01110100","01101010","01101010","10000001","11101101","11110001","11110101","11110111","11110101","11001000","01101001","01101100","01101011","01101100","01110111","01111111","10010001","10010001","10010001","10010001","10001110","10110000","11011100","11101110"),

("11110111","11111101","10111101","10001101","10010001","10010001","10010001","10010110","01110000","01001101","01001101","01011111","10110111","10111010","10111100","10111101","10111101","10011000","01001100","01010000","01001111","01010011","10001100","10010100","10010001","10010001","10010001","10010001","10001101","10111101","11111101","11110111"),

("11110110","11111010","10111100","10001101","10010001","10010001","10010001","10010001","10010010","10010011","10010101","10000101","00111100","00111001","00111001","00111001","00110111","01010101","10010110","10010011","10010011","10010011","10010001","10010001","10010001","10010001","10010001","10010001","10001101","10111100","11111010","11110110"),

("11110110","11110111","11101010","11100010","11011111","10011000","10001100","10010001","10010001","10010001","10010001","10001110","10000000","10000000","10000000","10000000","01111111","10000101","10010010","10010001","10010001","10010001","10010001","10010001","10001110","10001110","11010011","11100011","11101010","11110111","11110110"),

("11110110","11110110","11111000","11111010","11110111","11000000","10101100","10010001","10010001","10010001","10010001","10010001","10010011","10010011","10010011","10010011","10010011","10010010","10010001","10010001","10010001","10010001","10010001","10010001","10001111","10100101","10111000","11101101","11111010","11110111","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11111100","11100010","10001101","10001110","10001110","10010000","10010001","10010001","10010001","10010001","10010001","10010001","10010001","10010001","10010001","10001110","10001110","10001001","11001001","11111110","11111000","11111100","11111100","11111100","11111100"),

("11110110","11110110","11110110","11110110","11110110","11111000","11100100","10100100","10100101","10100101","10010101","10010000","10010001","10010001","10010001","10010001","10010001","10010001","10010001","10010001","10010001","10010010","10100011","10100101","10100010","11010001","11111001","11110111","11111111","11111111","11111111","11111111"),

("11110110","11110110","11110110","11110110","11110110","11110110","11110110","11110110","11110110","11110110","10100011","10001011","10001110","10001110","10001110","10001110","10001110","10001110","10001110","10001101","10010011","11101010","11111000","11110110","11110110","11110110","11111111","11111111","11111111"),

("11110110","11110110","11110110","11110110","11110110","11110110","11110110","11110110","11110110","11110110","11100000","11011001","11011010","11011010","11011010","11011010","11011010","11011010","11011010","11011010","11011011","11110011","11110110","11110110","11110110","11110110","11111111","11111111","11111111","11111111")

);

constant banana_green : ice_cream_column := (

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11111010","11001100","11000111","11001000","11000111","11000110","11001010","11111001","11111111","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11101111","01000110","00110100","00110111","00110111","00110101","01000000","11100111","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11111111","11111111","11111110","11111110","11111110","11111110","11111110","11111111","11101110","01000010","01000011","01101110","01111001","10000010","01111111","01100100","01011111","01011001","10110111","11111111","11111111","11111111","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111111","11101111","10110110","10110110","10110110","11110000","11111111","11111111","11111111","11111111","11110100","10001110","01111000","01100001","01110100","10000000","01111101","01011011","01010101","01010010","10001011","10111011","10110111","10110100","11010101","11111111","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11001100","00110101","00110111","00111001","11010001","11111101","11111000","11111000","11111100","11111110","11111111","11000011","00110011","01001100","01011001","01011011","10001010","10010010","10010101","01100000","00110011","00110101","00101101","10001010","11111111","11111110"),

("11111110","11111110","11111110","11111111","11111001","01101100","01011011","01111001","10011011","10111011","01101010","01010011","01010110","01010001","11001010","11111111","11111111","11000010","00110101","01001110","01011011","01011001","01001000","01001101","01100001","01101101","01110111","01000111","00110011","10001110","11111111","11111110"),

("11111110","11111110","11111110","11111111","11111000","01010000","01000110","10000110","10101101","11010010","01010100","00110001","00110101","00101111","11000001","11111111","11111111","11000010","00110001","01001100","01011100","01011001","00111101","01000010","01011001","01110001","10000100","01001001","00110010","10001110","11111111","11111110"),

("11111110","11111110","11111110","11111111","11111000","01010000","00111001","01010001","01111001","10011110","10100011","10100101","10100101","10100111","01111111","01101100","01101100","01111110","10100111","01111100","01100001","01100001","01000000","00111100","00111110","01000100","01001001","00111001","00101101","10001011","11111111","11111110"),

("11111110","11111110","11111110","11111111","11111010","10000111","01100100","00111010","01010110","01110001","10101100","10111110","10111100","10111111","01111011","01011011","01011010","10000001","11010110","10100011","10000100","10000011","01000010","01001000","01110011","01110001","01110000","01110010","01101100","10101101","11111111","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111111","11010011","00111010","00111101","00111111","01110101","10000101","10000011","10000001","10111000","11010010","11001111","11001110","11001011","11001100","11001110","11000110","01000111","01101011","11111111","11111111","11111111","11111111","11111111","11111111","11111110","11111110"),

("11111110","11111110","11111110","11111111","11111001","01110110","01110010","10110010","10110001","10101111","01011110","01000110","01001001","01000111","01111110","10011001","10011000","10000011","01010010","10011110","11010000","11001010","10110100","10011011","01011011","10111000","11111111","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111111","11111000","01001110","01010000","10110110","11000100","11010000","10000101","01101111","01110001","01110001","01101011","01101001","01101001","01101011","01101101","10101000","11010000","11001100","11010011","10101000","00101011","10011101","11111011","11111100","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111101","11111110","11110111","01010100","01001000","01111110","10100110","11001100","11001101","11001110","11010000","11010100","01101110","00111110","00111110","01101110","11010101","11001011","11001000","11001000","11001111","10101000","00110110","10010000","11011011","11110110","11111110","11111101","11111110","11111110"),

("11111110","11111111","11101101","11100000","11011011","01010010","00111011","01000011","01101011","10010001","10001101","10010111","11001010","11001100","10111111","10111000","10111000","10111111","11001111","10010110","01110000","01111000","11000011","11000000","10011100","01101011","01000101","11000101","11100010","11101101","11111111","11111110"),

("11111110","11111111","11101100","11011111","11011010","01001110","00110101","00110100","01011111","10001000","10000011","10010000","11001100","11001101","11001110","11001111","11010000","11001111","11010010","10001101","01011110","01101000","11000000","11000100","10101111","01100101","00101001","10111011","11011101","11101001","11111111","11111110"),

("11111110","11111111","10100011","01011001","01100010","10011100","10100110","10100111","01111000","01001011","01001101","01011011","10011010","10011011","10011011","10011000","01111011","01111000","01110110","10011001","10110001","10110000","11001010","11000100","10101110","01100110","00101100","10111101","11011110","11101010","11111111","11111110"),

("11111011","11111100","10000000","00100001","00101111","10101000","11000011","11010101","10011001","01100000","01100101","01100111","01101111","01101111","01101111","01110000","01110111","01111000","01110101","10101100","11010010","11001101","11000100","10111000","10100011","01100001","00101100","10111100","11011110","11100110","11110101","11111011"),

("11110100","11100000","01111000","00101000","00110011","01110110","10010101","11001100","11001101","11001110","11010001","10111000","01000010","00111110","00111101","01001010","11000011","11010000","11001110","11001100","11001011","11001001","10101100","10011110","10000111","01010101","00101110","10111101","11011111","11011011","11011001","11110100"),

("11110100","11011100","11000111","10110111","10110100","01001111","01010010","10010010","10110010","11010000","11001110","11001000","10110000","10101111","10101111","10110001","11001001","11001100","11001100","11001101","11001111","11001011","10010111","01111001","00111011","10000010","10111101","11010101","11011100","11011011","11011010","11110100"),

("11110100","11011010","11011101","11100000","11011001","01000100","00111100","01111011","10011000","10110100","10111011","11000001","11001111","11001111","11001111","11001111","11001100","11000111","10111110","10110111","10110010","10110000","10000101","01101001","00101001","10010000","11100110","11011100","11011011","11011011","11011010","11110100"),

("11110100","11011010","11011011","11011100","11010101","01000010","00111001","01110010","01111011","10000100","10100000","10101111","11001100","11001101","11001100","11001100","11001110","11000010","10101000","10010010","10000010","10000010","01110110","01100011","00110000","10010001","11100010","11011011","11011011","11011011","11011010","11110100"),

("11110100","11011010","11011011","11011100","11010111","01100101","01010101","01100101","01110100","10000010","10011010","10100110","10111100","10111101","11000010","11000100","10111111","10110100","10100001","10001110","10000000","10000000","01101000","01100001","01010110","10100001","11100000","11011011","11011011","11011011","11011010","11110100"),

("11110100","11011001","11011011","11011011","11011011","11100000","10110110","00110100","01010110","01110101","10000011","10000111","10000101","10000100","10011110","10100111","10001000","10000101","10001000","01111011","01110011","01110000","00101111","01010111","11011110","11011011","11011011","11011011","11011011","11011001","11110100"),

("11111011","11110101","11100110","11011010","11011011","11011101","11010010","10110010","01110011","00110111","01000000","01001011","01110111","01111000","10001010","10010000","01111100","01100111","01000001","00111110","00111011","01000001","10011101","10111001","11011101","11011100","11011011","11011011","11011010","11100110","11110101","11111011"),

("11111110","11111111","11101010","11011010","11011011","11011011","11011100","11100110","10011101","01011000","01011101","01011101","01011110","01011110","01100111","01101010","01011111","01011110","01011101","01011110","01011101","01100100","11010010","11100010","11011011","11011011","11011011","11011010","11101010","11111111","11111110"),

("11111110","11111111","11101010","11011010","11011011","11011011","110110
11","11011011","11011101","11011111","11100011","11000011","00110001","001
01101","00101110","00101110","00101000","01100100","11100100","11011111","
11011111","11011111","11011011","11011011","11011011","11011011","1101101
1","11011011","11011010","11101010","11111111","11111110"),

("11111110","11111110","11111010","11110111","11110110","11011110","110110
10","11011011","11011011","11011011","11011100","11010110","10111001","101
11000","10111000","10111000","10110111","11000011","11011100","11011011","
11011011","11011011","11011011","11011011","11011011","11011010","1101101
0","11110010","11110111","11111010","11111110","11111110"),

("11111110","11111110","11111111","11111111","11111110","11101011","111001
01","11011011","11011011","11011011","11011011","11011100","11011111","110
11111","11011111","11011111","11011111","11011110","11011011","11011011","
11011011","11011011","11011011","11011011","11011011","11011010","11100010","1110100
1","11111011","11111111","11111111","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111111","111101
11","11011010","11011010","11011010","11011011","11011011","11011011","110
11011","11011011","11011011","11011011","11011011","11011011","11011011","
11011011","11011011","11011010","11011010","11011000","11101110","1111111
1","11111110","11111111","11111111","11111111","11111111"),

("11111110","11111110","11111110","11111110","11111110","11111111","111110
00","11100010","11100010","11100010","11011100","11011011","11011011","110
11011","11011011","11011011","11011011","11011011","11011011","11011011","
11011011","11011011","11100001","11100010","11100001","11110001","1111111
1","11111110","11111111","11111111","11111111","11111111"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11100001","11011001","11011010","110
11010","11011010","11011010","11011010","11011010","11011010","11011010","
11011010","11011100","11111010","11111111","11111110","11111110","1111111
0","11111110","11111111","11111111","11111111","11111111"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11110110","11110100","11110100","111
10100","11110100","11110100","11110100","11110100","11110100","11110100","
11110100","11110100","11111101","11111110","11111110","11111110","1111111
0","11111110","11111111","11111111","11111111","11111111")

);

constant banana_blue : ice_cream_column := (

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11111001","10111111","10111001","10111010","10111001","10111000","10111100","11110111","11111111","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11101011","00010110","00000000","00000011","00000011","00000001","00001110","11100001","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11101010","00010011","00000001","00001111","00010101","00011001","00011011","00110110","00111011","00110010","10100110","11111111","11111111","11111111","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111111","11101011","10100100","10100100","10100100","11101101","11111111","11111111","11111111","11111111","11110010","01110001","01000110","00001011","00010111","00011110","00011100","00001100","00001010","00000011","01011111","10101100","10100101","10100001","11001011","11111111","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","10111110","00000000","00000000","00000001","11000101","11111100","11110110","11110110","11111100","11111110","11111111","10110100","00000001","00010100","00011100","00011011","00011101","00011101","00011111","00001110","00000000","00000000","01101100","11111111","11111110"),

("11111110","11111110","11111110","11111111","11110111","01000111","00100001","00010100","00100001","00101101","00101010","00101010","00101010","00100100","10111101","11111111","11111111","10110011","00000001","00010100","00011100","00011010","00001101","00010000","00011011","00011001","00010111","00001011","00000000","01110001","11111111","11111110"),

("11111110","11111110","11111110","11111111","11110110","00100101","00000100","00010111","00100110","00110011","00001100","00000010","00000011","00000000","10110010","11111111","11111111","10110011","00000000","00010011","00011100","00011010","00001011","00001101","00011011","00011011","00011010","00001100","00000000","01110001","11111111","11111110"),

("11111110","11111110","11111110","11111111","11110110","00100011","00000000","00000101","00010011","00100000","00100101","00100111","00100111","00100110","01000001","01001101","01001101","01000000","00100101","00011101","

00010111","00010110","00001011","00001000","00000110","00000101","0000010
1","00000010","00000000","01101100","11111111","11111110"),

("11111110","11111110","11111110","11111111","11111000","01101000","001110
01","00000001","00001001","00010001","00100110","00101100","00101011","001
01100","00010110","00001011","00001011","00011000","00110100","00100110","
00011101","00011101","00001001","00011010","01001111","01001110","0100110
1","01001101","01000110","10011000","11111111","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111111","110001
11","00000111","00000100","00000000","00010010","00010111","00010110","000
10101","00101001","00110011","00110010","00110010","00110001","00110010","
00110010","00110000","00000110","01000110","11111111","11111111","1111111
1","11111111","11111111","11111111","11111110","11111110"),

("11111110","11111110","11111110","11111111","11111000","01010110","001100
11","00101010","00101001","00100111","00001010","00000010","00000011","000
00010","00010101","00011110","00011110","00010111","00000110","00100001","
00110011","00110001","00101001","00101101","00110011","10100110","1111111
1","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111111","11110110","00100100","000001
11","00101001","00101110","00110010","00011000","00010000","00010001","000
10001","00001111","00001110","00001110","00001111","00010000","00100100","
00110010","00110001","00110011","00100110","00000000","10001100","1111111
1","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111111","11110110","00101010","000001
11","00010101","00100011","00110001","00110010","00110010","00110011","001
10101","00010000","00000000","00000000","00010000","00110101","00110010","
00110001","00110001","00110010","00100111","00000000","10000111","1111101
0","11111100","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111011","11111011","11110001","00101001","000001
01","00001010","00010011","00011100","00011011","00011110","00110001","001
10001","00101100","00101010","00101010","00101100","00110001","00110100","
00110110","00110110","00110010","00101101","00011111","00100111","0011000
0","11010101","11111110","11111011","11111110","11111110"),

("11111110","11111110","11111110","11111111","11110110","00101001","000000
11","00000111","00010000","00011000","00010110","00011011","00110001","001
10010","00110011","00110011","00110010","00110010","00110001","00110101","
00110111","00110111","00110010","00101110","00100110","00010110","0000110
0","11001110","11111101","11111011","11111110","11111110"),

("11111110","11111111","10011010","01001001","01001111","00101101","001001
10","00100111","00010101","00000100","00000100","00001001","00011110","000
11111","00011111","00100000","00101011","00101100","00101100","00110000","

00110011","00110011","00110010","00101110","00100101","00010111","0001000
0","11001110","11111101","11111011","11111110","11111110"),

("11111101","11111111","01101110","00000000","00000111","00100110","001011
01","00110100","00011111","00001011","00001101","00001101","00010000","000
10000","00010000","00010010","00101010","00101100","00101100","00101111","
00110001","00110001","00101110","00101010","00100001","00010101","0000111
1","11001110","11111101","11111010","11111100","11111101"),

("11111100","11111111","01110010","00000101","00001111","00010100","000111
00","00110001","00110010","00110010","00110011","00101010","00000001","000
00000","00000000","00000100","00101101","00110010","00110001","00110001","
00110001","00110000","00100110","00100000","00010111","00010001","0001000
1","11001110","11111101","11111000","11111000","11111100"),

("11111100","11111010","11011101","11001000","11000011","00101001","000100
00","00011011","00100111","00110011","00110010","00110000","00100111","001
00111","00100111","00101000","00110000","00110001","00110010","00110010","
00110010","00110001","00011100","00010111","00001001","01110101","1101000
1","11110000","11111001","11111000","11111000","11111100"),

("11111100","11111000","11111011","11111111","11110110","00101111","000011
01","00011010","00100001","00101001","00101011","00101101","00110010","001
10010","00110010","00110010","00110001","00101111","00101100","00101010","
00101000","00100111","00011100","00010110","00000010","10010001","1111111
1","11111001","11111000","11111000","11111000","11111100"),

("11111100","11111000","11111000","11111010","11110000","00101000","000010
10","00100011","00011100","00010101","00100001","00100111","00110010","001
10010","00110001","00110001","00110010","00101110","00100100","00011100","
00010101","00010111","00100010","00011010","00000000","10001001","1111111
1","11111000","11111000","11111000","11111000","11111100"),

("11111100","11111000","11111000","11111001","11110010","01011000","001100
11","00011100","00011010","00011001","00011111","00100011","00101100","001
01100","00101101","00101110","00101101","00101001","00100001","00011100","
00011001","00011001","00011100","00100100","00110001","10100001","1111111
1","11111000","11111000","11111000","11111000","11111100"),

("11111100","11111000","11111000","11111000","11111000","11111111","110000
00","00000000","00010010","00100010","00011001","00010111","00010111","000
10111","00100000","00100011","00011000","00010111","00010111","00011101","
00100010","00100000","00000100","01000101","11111100","11111000","1111100
0","11111000","11111000","11111000","11111000","11111100"),

("11111101","11111100","11111010","11111000","11111000","11111011","111010
10","10111100","01011110","00000101","00001010","00001101","00011111","000
11111","00011100","00011011","00011111","00011001","00001001","00001011","

00001011","00010100","10100001","11001011","11111010","11111001","1111100
0","11111000","11111000","11111010","11111100","11111101"),

("11111110","11111110","11111011","11111000","11111000","11111000","111110
10","11111111","10100100","01000111","01010001","01001000","00011100","000
11011","00010101","00010011","00011001","00101011","01010010","01001111","
01001101","01010111","11101011","11111111","11111000","11111000","1111100
0","11111000","11111000","11111011","11111110","11111110"),

("11111110","11111110","11111011","11111000","11111000","11111000","111110
00","11111000","11111011","11111110","11111111","11011001","00010011","000
01101","00001110","00001110","00000110","01011000","11111111","11111110","
11111110","11111110","11111001","11111000","11111000","11111000","1111100
0","11111000","11111000","11111011","11111110","11111110"),

("11111110","11111110","11111101","11111101","11111101","11111001","111110
00","11111000","11111000","11111000","11111001","11110001","11001010","110
01001","11001001","11001001","11000111","11010111","11111001","11111000","
11111000","11111000","11111000","11111000","11111000","11111000","1111100
0","11111100","11111101","11111101","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111011","111110
01","11111000","11111000","11111000","11111000","11111001","11111110","111
11110","11111110","11111110","11111110","11111100","11111000","11111000","
11111000","11111000","11111000","11111000","11111000","11111001","1111101
0","11111101","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
01","11111000","11111000","11111000","11111000","11111000","11111000","111
11000","11111000","11111000","11111000","11111000","11111000","11111000","
11111000","11111000","11111000","11111000","11111000","11111011","1111111
0","11111110","11111111","11111111","11111111","11111111"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
01","11111001","11111001","11111001","11111000","11111000","11111000","111
11000","11111000","11111000","11111000","11111000","11111000","11111000","
11111000","11111000","11111001","11111001","11111001","11111100","1111111
0","11111110","11111111","11111111","11111111","11111111"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11111001","11111000","11111000","111
11000","11111000","11111000","11111000","11111000","11111000","11111000","
11111000","11111000","11111101","11111110","11111110","11111110","1111111
0","11111110","11111111","11111111","11111111","11111111"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11111100","11111100","11111100","111
11100","11111100","11111100","11111100","11111100","11111100","11111100","

11111100","11111100","11111110","11111110","11111110","11111110","1111111
0","11111110","11111111","11111111","11111111","11111111")

);

----------grape----------

constant grape_red :ice_cream_column := (

("11110110","11110110","11110110","11110110","11110110","11110110","111101
10","11110110","11110110","11110110","11110111","11110000","10010000","100
00010","01111111","01111111","01111111","01111111","01111101","01111110","
10000001","10100101","11111001","11110110","11110110","11110110","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","111101
10","11110110","11110110","11110110","11111000","11101010","00110011","001
01001","01010111","01010110","01010110","01010110","01100101","01100010","
00011101","01011011","11111100","11110110","11110110","11110110","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","111101
10","11110110","11110110","11110110","11111000","11101001","00100110","001
11001","11001101","11001100","11001011","11001011","11110111","11101010","
00011011","01010010","11111100","11110110","11110110","11110110","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","111101
10","11110110","11110110","11110110","11111000","11101001","00100011","001
01110","10101001","10101001","10100111","10100110","11000111","10111011","
00010100","01001111","11111100","11110110","11110110","11110110","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","111101
10","11110111","11110111","11110111","11111001","11101100","01001011","010
00001","01100110","01100101","01101111","01110111","01101001","01011111","
00110110","01101110","11111100","11110111","11110111","11110111","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","111101
01","11110100","11110100","11110100","11110100","11110100","11110111","110
00010","00010110","00010011","01001111","01111110","00101010","00101010","
11100101","11110100","11110100","11110100","11110100","11110100","1111011
0","11110110","11110110","11110110","11110110","11110110"),

("11110110","11110110","11111011","11111110","11111110","11111111","11010010","01001111","01010101","01011000","01010111","01011101","10110101","10011000","00010011","00010000","01001101","01111110","00101000","00100010","10110100","10100000","01010101","01011000","01010110","01011001","11100011","11111111","11111110","11111011","11110110","11110110"),

("11110110","11111010","10101110","10000100","10001000","10000111","10001101","10011111","01110110","01001100","01001111","01010000","01100101","01010100","00010101","00010010","01001110","01111110","00101010","00011011","01100001","01100000","01001110","01001110","01100110","01110110","10000101","10001000","10000100","10101110","11111010","11110110"),

("11111000","11111111","01110111","00101000","00101101","00100110","01011111","11111111","10101110","01011110","01100100","01100000","00111001","00101100","00010100","00010000","01001101","01111110","00101001","00010011","00101100","00111110","01100100","01100000","10010001","10101100","01000000","00101001","00100011","01110100","11111111","11111000"),

("11000010","01110011","01100000","01010110","10100010","10110010","10100111","10001110","10100111","11000000","11001001","11000110","01100001","01001110","00111100","00111001","01011111","01111110","01010010","01010010","10101000","10111000","11001101","11001111","10011000","01111001","10101001","10110101","11000000","10100101","01101111","11000010"),

("10100101","00100111","01100101","10000111","11100001","11111000","11001100","01010111","10100101","11110101","11111010","11110110","10001110","01110101","01010001","01010000","01011110","01101001","01111011","10001101","11101010","11110101","11111100","11111111","10011010","01011110","11000010","11011101","11111111","10110110","00100010","10100101"),

("10101000","00101100","10000001","10101101","11010110","11100010","11000010","01101010","10100110","11100011","11011110","11011010","10101111","10010101","01010010","01010011","01010010","01010000","10010100","10110000","11011011","11011110","11011101","11100001","10011000","01101101","10010000","10100011","11100011","10100101","00101010","10101000"),

("10101001","00101111","01111111","10101000","10100101","10100101","10100100","10100011","10100100","10100110","10100110","10100110","10100110","10010110","01100100","01100101","01100101","01100100","10010110","10100110","10100110","10100110","10100110","10100110","10100100","10100010","01101111","01101111","10101000","10000000","00101111","10101001"),

("11101100","11100011","01111101","01000100","01100110","01101100","01101011","01101000","10001000","10101010","10101001","10100110","01110111","01101000","01001101","01001101","01001100","01001100","01101000","01110111","10100101","10101001","10101001","10101011","10000010","01101001","01100010","01011011","01000100","01111101","11100011","11101100"),

("11001010","10001001","01100101","01010001","10100100","10110100","10110101","11000001","10100000","01111110","01101101","01100100","01100011","01110100","10101010","10101001","10110000","10110110","01110110","01100100","01110010","01101110","01100101","01100010","10010110","10110101","10111111","10101111","01010001","01100101","10001001","11001010"),

("10100101","00100111","01010011","01101011","11011010","11110000","11110010","11111111","10110010","01011011","01000001","00110110","01011100","10000011","11110000","11101111","11111000","11111111","10000110","01011101","01001110","01000100","00110000","00101010","10101001","11110110","11111111","11101001","01101011","01010011","00100111","10100101"),

("10101000","00101110","01110010","10010110","11011111","11101110","11101110","11110110","11000011","10001111","01001111","00111001","10000110","10100110","11101101","11101100","11110000","11110011","10100111","10001000","01010000","01011100","10000011","01111110","11000101","11110000","11110011","11100100","10010111","01110010","00101110","10101000"),

("10101000","00101101","10000010","10101110","11001000","11001110","11001101","11001110","10111010","10101000","01100100","01001101","10011101","10110001","11001101","11001101","11001101","11001100","10110000","10011110","01011010","01101110","10101011","10101000","11000000","11001110","11001101","11000111","10101111","10000010","00101101","10101000"),

("10101000","00101101","01111001","10011111","10100010","10100011","10100011","10100011","10100100","10100111","01111000","01101001","10011111","10100101","10100010","10100010","10100010","10100010","10100101","10011111","01101000","01111000","10100111","10100101","10100100","10100011","10100011","10100001","10010111","01110100","00101110","10101000"),

("10101000","00110000","01010110","01101001","10011010","10100100","10100011","10100100","10100100","10100110","01110110","01100110","10011110","10100101","10100101","10100101","10100101","10100101","10011110","01100110","01110110","10100110","10100100","10100100","10100100","10100101","10010100","00111100","00111001","00110010","10101000"),

("11000111","10000001","01010111","00111101","01001001","01001010","01010101","01110010","01110001","01110001","01001100","01000000","01101001","01111100","10100111","10100110","10100110","10100111","01111100","01101001","00111111","01001100","01110001","01110000","01110001","01110000","01010001","01000111","00110110","01010010","10000001","11000111"),

("11001101","10010010","01011011","00111010","00111001","00110111","01000011","01100100","01100100","01100110","01001011","01000011","01100111","01111001","10100001","10100001","10100001","10100001","01111000","01100111","01000100","01001100","01100110","01100101","01100100","01100010","00111111","00111000","00111011","01011100","10010010","11001101"),

("11001100","10001101","10001100","10000110","01010100","01001010","01001101","01010001","01011011","01100010","10110111","11011100","11101111","11010010","01101001","01101010","01101010","01101001","11000011","11100000","11101110","11000101","01100000","01100101","01011001","01010001","01001011","01010100","10000110","10001100","10001101","11001100"),

("11001011","10001000","10010000","10010010","10000000","01111110","01101101","00111011","01101001","10010101","11010101","11110001","11110110","11011111","10010010","10010011","10010011","10010010","11011010","11110001","11111000","11011010","10010001","10010111","01011110","00111110","01110011","10000001","10010010","10010000","10001000","11001011"),

("11011110","10111010","10100000","10001111","10010011","10010110","01111011","00110000","01100001","10010000","10111101","11010001","11010000","10111111","10001001","10001010","10001001","10001000","11000000","11010001","11010000","10111100","10001100","10010001","01010110","00110100","10000100","10010101","10001111","10100000","10111010","11011110"),

("11111000","11111110","10110110","10001101","10010001","10010011","01111110","01000110","01010010","01011011","10000001","10010001","10010011","10001110","01111010","01111010","01111100","01111110","10001110","10010011","10010001","10000001","01011011","01011101","01001111","01001000","10000101","10010011","10001101","10110110","11111110","11111000"),

("11110110","11111010","10110010","10001010","10010001","10010001","10010001","10010100","01100101","00110011","00111001","00111011","01000111","01101111","11101001","11101000","11110100","11111110","01110011","01000110","00111011","00111001","00110111","00110100","01101111","10010011","10010001","10010001","10001010","10110010","11111010","11110110"),

("11110110","11111001","11000101","10100111","10010100","10001111","10010000","10010100","01110001","01001101","00111110","00111010","01011101","10000011","11101011","11101010","11110011","11111100","10000110","01011101","00111010","00111110","01010000","01001100","01111001","10010100","10010000","10010100","10100111","11000101","11111001","11110110"),

("11110110","11110110","11111000","11110110","10100101","10010101","10010101","10010001","10010010","10010100","01010101","01000001","10010101","10110000","11100110","11100101","11100101","11100110","10110000","10010101","01000001","01010101","10010100","10010010","10010001","10010001","10010100","10100101","11110110","11111000","11110110","11110110"),

("11110110","11110110","11110110","11110110","11101101","11101111","11010100","10001100","10001110","10001110","10001010","10000100","01000101","01011000","10110000","10101111","10101111","10110000","01011000","01000101","10000100","10001010","10001110","10001110","10001101","10001111","11011101","11110000","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11111000","11111001","11110000","11010110","11010111","11011001","10100011","10001010","01110100","01101001","01010010","01010010","01010010","01010010","01101001","01110100","10001010","10100011","11011001","11010111","11010111","11011000","11110011","11111000","11110110","11110110","11110110","11110110"),

("11110110","11110110","11110110","11110110","11110110","11110110","11110111","11111010","11111010","11111100","11001101","10111010","11000000","10110010","10000100","10000101","10000101","10000100","10110010","11000000","10111010","11001101","11111100","11111010","11111010","11111010","11110111","11110110","11110110","11110110","11110110")

);

constant grape_green :ice_cream_column := (

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11111010","10111001","10101111","10101011","10101011","10101011","10101011","10101010","10101011","10101110","11000111","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11110101","01101010","01100110","10011000","10010111","10010111","10010111","10011010","10010100","01011010","10001001","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11110011","01000010","01011100","11111100","11111011","11111010","11111010","11111111","11101011","00111001","01101010","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11110011","01000001","01010101","11100011","11100010","11011111","11011110","11101001","11010111","00110101","01101001","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11111111","11111111","11111111","11110110","01100101","01100101","10100110","10100100","10110001","10111100","10101011","10011101","

01010110","10000101","11111111","11111111","11111111","11111111","1111111
0","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
01","11111100","11111100","11111100","11111100","11111100","11111111","110
01111","00110110","00110010","10000011","11000100","01010010","01000111","
11101110","11111100","11111100","11111100","11111100","11111100","1111111
0","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111111","11111111","11111111","11111111","110011
01","00111001","00111110","00111111","00111110","01000100","10001111","100
00001","00110100","00101111","10000010","11000100","01010001","00111011","
10010010","01111110","00111101","00111111","00111110","01000010","1110000
0","11111111","11111111","11111111","11111110","11111110"),

("11111110","11111111","10100111","01110011","01111000","01111001","011011
00","01001000","00111011","00101101","00101110","00101111","00111101","001
11100","00110110","00110000","10000010","11000100","01010011","00110011","
00111110","00111010","00101110","00101110","00110011","00110111","0110110
1","01111001","01110011","10100111","11111111","11111110"),

("11111110","11111111","01100010","00000101","00001101","00001010","001010
00","01111011","01011111","01000100","01000101","01000001","00011000","000
11011","00110110","00110000","10000010","11000100","01010011","00101110","
00010010","00100000","01000101","01000011","01010000","01010111","0001100
1","00001011","00000001","01011111","11111111","11111110"),

("10111111","01011111","01000110","00110110","01010001","01010110","010101
01","01010011","01011100","01100011","10001101","10010111","01000001","001
10100","00110001","00101011","10000011","11001010","01011100","00111110","
01010010","01101001","10011101","10011110","01101000","01001000","0101001
1","01011110","10001100","01111110","01011100","10111111"),

("10011100","00000101","00111000","01010000","01110000","01111000","011010
01","01000001","01011011","01110101","10011001","10100011","01010110","010
00101","00101110","00101010","01011110","10000111","01010111","01001111","
01110010","10000100","10100111","10101010","01101000","01000010","0110001
0","01110011","10101011","01110011","00000001","10011100"),

("10011111","00001100","01000000","01011000","01101001","01101111","011001
00","01000110","01011010","01101111","01101011","01101001","01011000","010
01101","00110011","00110011","00110000","00101100","01001100","01011000","
01101100","01101100","01101010","01101011","01010100","01000111","0100111
0","01010100","01101101","01001110","00001011","10011111"),

("10100001","00010000","01000000","01010101","01010100","01010100","010101
00","01010100","01010100","01010100","01010100","01010100","01010100","010
10001","01000101","01000101","01000101","01000101","01010001","01010100","

01010100","01010100","01010100","01010100","01010100","01010100","0100100
0","01001000","01010110","01000001","00010000","10100001"),

("11110010","11100110","01100011","00011011","01000001","01000111","010001
01","01000010","01001011","01010101","01010110","01010101","01001010","010
00011","00110000","00110000","00101110","00101101","01000010","01001010","
01010100","01010101","01010110","01010111","01001101","01000110","0100001
1","00111100","00011011","01100011","11100110","11110010"),

("11001000","01111001","01001100","00110000","01011001","01100000","011011
00","10010010","01101110","01001010","00111000","00110001","01000100","010
01010","01010110","01010100","01110000","10000111","01010100","01000100","
00111111","00111010","00110001","00101111","01001111","01100011","1000100
1","10000010","00110001","01001011","01111001","11001000"),

("10011100","00000100","00110100","01001010","01101101","01110010","100010
01","11001110","10001000","01000001","00100011","00011000","01000000","010
10001","01110101","01110010","10100100","11001101","01100011","01000000","
00101111","00100110","00010100","00010001","01010001","01111001","1011110
0","10110110","01001100","00110011","00000100","10011100"),

("10100000","00001101","00111101","01010011","01101110","01110011","011110
01","10001111","01101111","01001111","00100110","00011000","01001010","010
11000","01110011","01110010","10000001","10001110","01011110","01001011","
00101111","00110011","01000010","00111111","01100001","01110101","1000100
1","10000100","01010011","00111100","00001101","10100000"),

("10100000","00001101","01000000","01011000","01100100","01100110","011001
01","01100100","01011100","01010100","00110110","00101100","01010000","010
11001","01100110","01100110","01100100","01100011","01011000","01010001","
00111001","01000000","01010111","01010101","01100000","01100111","0110010
0","01100001","01011001","01000001","00001101","10100000"),

("10011111","00001100","00111110","01010100","01010011","01010011","010100
11","01010011","01010011","01010100","01001011","01001000","01010011","010
10100","01010011","01010011","01010011","01010011","01010100","01010011","
01000111","01001011","01010100","01010100","01010011","01010011","0101001
1","01010010","01001100","00111001","00001101","10011111"),

("10100001","00010001","00110111","01000111","01010001","01010011","010100
11","01010100","01010100","01010100","01001001","01000101","01010011","010
10100","01010100","01010100","01010100","01010100","01010100","01010011","
01000101","01001001","01010100","01010100","01010100","01010100","0101010
0","01001011","00011011","00011010","00010011","10100001"),

("11100010","10111110","01010101","00011000","00100001","00100001","001011
00","01001010","01001001","01001001","00101010","00100000","01000011","010
01010","01010100","01010100","01010100","01010100","01001011","01000100","

00100000","00101010","01001001","01001000","01001001","01001000","0010100
0","00100000","00010001","01010000","10111110","11100010"),

("11110001","11100101","01100101","00011100","00011010","00010111","001001
00","01000101","01000101","01000111","00101000","00011110","01001000","010
01111","01010011","01010011","01010011","01010011","01001011","01000100","
00100011","00101100","01000111","01000110","01000101","01000011","0010000
0","00011010","00011101","01100101","11100101","11110001"),

("11110000","11011010","11001111","11000011","00111110","00100011","001010
10","00110001","00111100","01000110","01011110","01101110","10110010","100
11111","01000110","01000111","01000111","01000111","01100101","01110110","
10110001","10010100","01000011","01000111","00111001","00110001","0010010
1","00111110","11000011","11001111","11011010","11110000"),

("11101111","11011000","11011101","11011101","10101101","10101000","100000
11","00011011","00110111","01010001","01101010","01111000","10001101","100
00001","01001111","01010000","01010000","01010000","01101100","01111000","
10001111","01111101","01001110","01010010","00110001","00100000","1000111
1","10110000","11011101","11011101","11011000","11101111"),

("11110110","11101010","11100000","11011011","11100000","11100111","101011
01","00001101","00110001","01010010","01100001","01101000","01100101","010
11111","01001110","01001110","01001101","01001100","01100000","01100111","
01100101","01011110","01001111","01010010","00101001","00010101","1100000
0","11100101","11011011","11100000","11101010","11110110"),

("11111110","11111111","11101000","11011010","11011011","11100000","101100
10","00110111","00111100","00111110","01000101","01001001","01001100","010
01100","01001101","01001100","01010101","01011100","01001111","01001011","
01001001","01000101","00111110","00111110","00111011","00111101","1100000
1","11011110","11011010","11101000","11111111","11111110"),

("11111110","11111111","11100111","11011001","11011011","11011011","110110
10","11100010","01111001","00001011","00011000","00011011","00101010","001
11101","01110011","01101111","10011111","11000110","01001110","00101000","
00011011","00011000","00010100","00001100","10010001","11100001","1101101
1","11011011","11011001","11100111","11111111","11111110"),

("11111110","11111111","11101101","11100011","11011100","11011011","110110
11","11100010","10010101","01000100","00100101","00010111","00110100","010
00110","01110100","01110001","10010100","10110001","01010011","00110011","
00010111","00100101","01001010","01000011","10100110","11100010","1101101
1","11011100","11100011","11101101","11111111","11111110"),

("11111110","11111110","11111111","11111110","11100010","11011100","110111
00","11011011","11011101","11100011","01011000","00100001","01001100","010
11000","01110000","01110000","01110000","01101111","01011000","01001100","

00100001","01011000","11100011","11011110","11011100","11011011","1101110
0","11100010","11111110","11111111","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111011","11111100","111100
11","11011001","11011010","11011010","11001111","10111110","00101000","001
00100","01011010","01011001","01011001","01011010","00100100","00101000","
10111110","11001111","11011010","11011010","11011010","11011011","1111011
0","11111100","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111111","11111111","111111
00","11110011","11110011","11110100","11100011","11011000","10100111","100
00001","00100000","00100001","00100001","00100000","10000001","10100111","
11011000","11100011","11110100","11110011","11110011","11110011","1111110
1","11111111","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111111","11111111","11111111","11110000","11101010","11110100","110
10111","01110111","01111000","01111000","01110111","11010111","11110100","
11101010","11110000","11111111","11111111","11111111","11111111","1111111
0","11111110","11111110","11111110","11111110","11111110")

);

constant grape_blue :ice_cream_column := (

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11111111","11110111","10010010","100
00011","10000001","10000001","10000001","10000001","01111111","01111111","
10000010","10101000","11111111","11111110","11111110","11111110","1111111
0","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11111111","11110001","00110001","001
00100","01001110","01001110","01001110","01001110","01100001","01011111","
00011010","01011011","11111111","11111110","11111110","11111110","1111111
0","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11111111","11110001","00100110","001
10101","10111110","10111100","10111100","10111011","11110100","11101010","
00011001","01010011","11111111","11111110","11111110","11111110","1111111
0","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11110001","00100100","00101000","10001111","10001110","10001101","10001101","10110100","10101100","00010010","01010001","11111111","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111110","11111111","11111111","11111111","11111111","11110100","01001110","00111010","01000011","01000011","01001001","01001110","01000011","00111111","00110101","01110010","11111111","11111111","11111111","11111111","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","11111101","11111100","11111100","11111100","11111100","11111100","11111111","11001000","00010100","00010011","00110111","01010100","00011111","00101010","11101100","11111100","11111100","11111100","11111100","11111100","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111111","11111111","11111111","11111111","11010100","01010010","01011000","01011010","01011001","01011111","10101000","10001100","00010100","00010010","00110111","01010100","00100000","00100010","10100110","10010111","01011000","01011010","01011000","01011011","11100100","11111111","11111111","11111111","11111110","11111110"),

("11111110","11111111","10110011","10000111","10001011","10001011","10001000","10000011","01100101","01000110","01001000","01001001","01011101","01001110","00010100","00010010","00110111","01010100","00100001","00011010","01011010","01011000","01000111","01000111","01011000","01100100","10000100","10001100","10000111","10110011","11111111","11111110"),

("11111110","11111111","01111000","00101000","00101101","00101000","01010100","11001110","10010001","01010011","01011000","01010101","00111000","00101100","00010011","00010000","00110110","01010100","00100000","00010100","00101110","00111100","01011000","01010101","01111001","10001101","00111100","00101011","00100011","01110101","11111111","11111110"),

("11000111","01110101","01011100","01001101","10001000","10010100","10001100","01111000","10001011","10011110","10110100","10110110","01011000","01000111","00111000","00110111","01000110","01010010","01000100","01001001","10001100","10011110","10111100","10111110","10000111","01100110","10001101","10011001","10110001","10011101","01110001","11000111"),

("10101010","00100111","01011000","01110001","10110110","11001000","10100111","01001110","10001010","11000110","11010110","11010111","01111001","01100100","01001010","01001010","01001100","01001101","01100110","01111000","10111110","11001011","11011101","11100001","10000111","01010011","10011110","10110110","11100010","10100010","00100011","10101010"),

("10101100","00101101","01101100","10001101","10101110","10110111","10011111","01011101","10001010","10111000","10110011","10110000","10001111","01111011","01001011","01001100","01001100","01001011","01111011","10001111","10110010","10110100","10110010","10110101","01111111","01011111","01111000","10000110","10110110","10001000","00101011","10101100"),

("10101110","00110001","01101100","10001001","10000111","10000111","10000111","10000110","10000111","10001000","10001000","10001000","10000111","01111100","01011010","01011010","01011010","01011010","01111100","10000111","10001000","10001000","10001000","10001000","10000111","10000101","01100001","01100001","10001001","01101101","00110001","10101110"),

("11110100","11101010","01111101","01000000","01011001","01011110","01011101","01011010","01110010","10001011","10001010","10001000","01100111","01011100","01000111","01000111","01000110","01000110","01011011","01100111","10000111","10001010","10001010","10001100","01101110","01011100","01010110","01010010","00111111","01111101","11101010","11110100"),

("11010000","10001100","01100010","01001010","10001000","10010100","10011001","10101111","10001101","01101010","01011110","01011000","01011000","01100100","10001101","10001100","10011011","10101000","01101010","01011000","01100010","01100000","01011001","01010111","01111110","10010110","10101011","10011111","01001011","01100010","10001100","11010000"),

("10101001","00100111","01001011","01011110","10110010","11000010","11001101","11110110","10100101","01010001","00111111","00110111","01010011","01110000","11000011","11000000","11011100","11110010","01111010","01010011","01001000","01000000","00110010","00101101","10001101","11000111","11101010","11011001","01011111","01001011","00100111","10101001"),

("10101101","00101110","01100010","01111101","10110101","11000000","11000011","11010001","10100100","01110111","01001010","00111001","01110001","10001001","11000000","10111111","11000111","11001110","10001100","01110010","01001001","01010010","01101111","01101011","10100001","11000010","11001100","11000001","01111101","01100010","00101110","10101101"),

("10101101","00101110","01101110","10001110","10100011","10101000","10100110","10100110","10010111","10001010","01011001","01001001","10000010","10010000","10100111","10100111","10100110","10100101","10010000","10000011","01010001","01100000","10001100","10001001","10011100","10101000","10100110","10100001","10001110","01101110","00101110","10101101"),

("10101100","00101101","01100111","10000011","10000100","10000101","10000101","10000101","10000110","10001000","01101000","01011101","10000011","10000111","10000101","10000101","10000101","10000101","10000111","10000011","01011100","01100111","10001000","10000111","10000110","10000101","10000101","10000011","01111101","01100011","00101101","10101100"),

("10101110","00110011","01001111","01011100","10000000","10000111","100001
10","10000110","10000110","10000111","01100101","01011010","10000010","100
00111","10000111","10000111","10000111","10000111","10000111","10000010","
01011010","01100101","10000111","10000110","10000110","10000110","1000100
0","01111011","00111100","00111010","00110100","10101110"),

("11101111","11011101","01110011","00110110","01000101","01000101","010011
01","01100011","01100010","01100010","01000111","00111110","01011100","011
01001","10000111","10000111","10000111","10000111","01101001","01011100","
00111110","01000111","01100010","01100001","01100010","01100010","0100101
1","01000011","00110001","01101111","11011110","11101111"),

("11111101","11111111","10000100","00111100","00111010","00110111","010000
00","01011001","01011001","01011010","01000111","01000001","01011101","011
01001","10000100","10000100","10000100","10000100","01100111","01011011","
01000011","01001000","01011010","01011001","01011001","01010111","0011110
1","00111001","00111101","10000100","11111111","11111101"),

("11111100","11111000","11101101","11100001","01011010","00111110","010001
00","01001011","01010010","01010111","10010111","10110101","11011001","101
11111","01011100","01011101","01011101","01011100","10100000","10111001","
11011001","10110100","01010110","01011001","01010000","01001010","0100000
0","01011010","11100001","11101101","11111000","11111100"),

("11111100","11111000","11111010","11111010","11001001","11000100","101000
00","00111001","01011011","01111100","10101101","11000100","11010000","101
11101","01111001","01111010","01111010","01111001","10110001","11000100","
11010001","10111000","01111001","01111101","01010100","00111110","1010101
1","11001100","11111010","11111010","11111000","11111100"),

("11111101","11111011","11111001","11111000","11111101","11111111","110010
10","00101101","01010100","01111001","10011010","10101001","10101000","100
11011","01110100","01110100","01110011","01110010","10011101","10101001","
10101001","10011010","01110110","01111010","01001011","00110101","1101111
0","11111111","11111000","11111001","11111011","11111101"),

("11111110","11111110","11111010","11111000","11111000","11111101","110100
00","01010111","01010110","01010001","01101101","01111000","01111010","011
10110","01101001","01101001","01101101","01110001","01111000","01111010","
01111000","01101101","01010001","01010010","01010110","01011101","1101111
1","11111011","11111000","11111010","11111110","11111110"),

("11111110","11111110","11111010","11111000","11111000","11111000","111101
11","11111110","10011000","00101011","00111000","00111011","01000011","011
00001","10111101","10111010","11010101","11101010","01101011","01000010","
00111011","00111000","00110100","00101100","10101111","11111110","1111100
0","11111000","11111000","11111010","11111110","11111110"),

("11111110","11111110","11111011","11111001","11111000","11111000","111110
00","11111111","10110011","01100100","01000101","00110111","01010011","011
01111","10111111","10111101","11010000","11100001","01110110","01010010","
00110111","01000101","01101010","01100011","11000100","11111111","1111100
0","11111000","11111001","11111011","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111001","11111000","111110
00","11111000","11111010","11111111","01110111","01000010","01111011","100
10000","10111011","10111010","10111010","10111001","10010000","01111100","
01000010","01110111","11111111","11111011","11111001","11111000","1111100
0","11111001","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111101","11111101","111111
00","11111000","11111000","11111001","11101100","11011011","01001001","010
01011","10010000","10001111","10001111","10001111","01001011","01001001","
11011011","11101100","11111001","11111000","11111000","11111000","1111110
0","11111101","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
01","11111100","11111100","11111100","11111100","11110111","11000101","101
00010","01000111","01001000","01001000","01000111","10100010","11000101","
11110111","11111100","11111100","11111100","11111100","11111100","1111111
0","11111110","11111110","11111110","11111110","11111110"),

("11111110","11111110","11111110","11111110","11111110","11111110","111111
10","11111110","11111110","11111110","11111100","11111011","11111111","111
00110","10001001","10001010","10001010","10001001","11100110","11111111","
11111011","11111100","11111110","11111110","11111110","11111110","1111111
0","11111110","11111110","11111110","11111110","11111110")

);

constant snow_red :snow_tile := (

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

```
("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11110","11110","11110","11110","11110","11110","11110","11110","11110","111
10","11110","11110","11110","11110","11110","11110","11110","11110","11110","
```

11110","11110","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11110","11110","11110","11110","11110","111
11","10111","10010","10011","10011","10011","10011","10011","10010","
10111","11111","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11110","11110","11110","11110","11110","111
00","10101","10001","10001","10001","10001","10001","10001","10001","
10101","11100","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11110","11110","11110","11111","11011","100
00","10100","10110","10101","10101","10101","10101","10101","10110","
10011","10000","11100","11111","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11110","11110","11110","11001","10110","101
01","10111","11000","11000","11000","11000","11000","11000","11000","
10111","10101","10110","11001","11111","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11111","11111","11111","10011","10010","110
10","11010","11010","11010","11010","11010","11010","11010","11010","11010","
11010","11010","10001","10100","11111","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11101","11101","11101","10011","10010","110
01","11001","11001","11001","11010","11010","11010","11010","11010","11010","
11010","11010","10010","10100","11101","11101","11101","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11111","11111","10011","10001","10001","10001","10001","100
01","10100","10110","10111","11010","11101","11110","11110","11011","11010","
11010","11010","10010","10000","10001","10101","10110","11001","11111","1111
1","11111","11110","11110"),

("11110","11101","11001","11001","10100","10010","10011","10011","10011","100
11","10011","10011","10110","11010","11101","11110","11111","11000","10110","
10111","11000","10100","10011","10011","10101","10101","11000","11101","1101
1","11100","11110","11110"),

("11111","11011","10000","10000","10101","10110","10110","10110","10110","101
10","10010","10000","10100","11010","11101","11110","11111","10100","10000","
10010","10110","10110","10110","10110","10110","10101","10111","11001","1011
0","11000","11111","11110"),

("11001","10111","10111","10111","11000","11000","11000","10110","10101","101
01","10101","10100","10111","11110","11110","11110","11111","10111","10100","
10101","10110","11000","11001","11000","11000","11000","11001","11001","1100
1","11001","11011","11101"),

("11000","10111","11011","11011","11010","11010","11010","11000","10111","101
11","10110","10110","11001","11110","11110","11110","11111","11001","10110","
10110","10110","11010","11010","11010","11010","11010","11010","11010","1101
0","11010","11010","11101"),

("11010","11010","11110","11110","11110","11110","11110","11110","11110","111
10","11011","11001","11000","10101","11101","11111","11110","11011","11001","
11001","11001","11101","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11100","11100","11110","11110","11110","11110","11110","11110","11110","111
10","11101","11100","11010","10111","11011","11101","11110","11100","11100","
11100","11100","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11110","11110","11110","11110","11110","111
10","11110","11111","11101","11010","11001","11011","11110","11110","11111","
11111","11111","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11110","11110","11110","11110","11110","111
10","11110","11110","11101","11010","11010","11011","11110","11110","11110","
11110","11110","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11110","11110","11110","11110","11110","11110","11110","11110","11110","111
10","11110","11110","11110","11110","11110","11110","11110","11110","11110","
11110","11110","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11110"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","

```
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111")

);



   constant snow_green : snow_tile := (

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),
```

("11111","11111"),

("11111","11111"),

("11111","11111"),

("11111","11111"),

("11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","11111","11100","11010","11010","11010","11010","11010","11010","11010","11010","11100","11111","11111","11111","11111","11111","11111","11111","11111","11111","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","11110","11011","11010","11010","11010","11010","11010","11010","11010","11010","11100","11110","11111","11111","11111","11111","11111","11111","11111","11111","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11110","11001","11100","11101","11101","11101","11101","11101","11101","11101","11101","11011","11001","11110","11111","11111","11111","11111","11111","11111","11111","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11101","11100","11100","11101","11110","11110","11110","11110","11110","11110","11110","11110","11101","11100","11100","11101","11111","11111","11111","11111","11111","11111","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11011","11010","11111","11110","11110","11110","11110","11110","11110","11110","11110","11110","

11110","11111","11010","11011","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11011","11010","111
10","11110","11110","11110","11110","11110","11110","11110","11110","11110","
11110","11110","11010","11011","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11011","11010","11010","11001","11001","110
01","11100","11101","11110","11110","11111","11111","11111","11111","11111","
11110","11110","11010","11001","11010","11100","11101","11110","11111","1111
1","11111","11111","11111"),

("11111","11111","11101","11101","11011","11011","11011","11011","11011","110
11","11011","11011","11100","11110","11111","11111","11111","11101","11100","
11101","11110","11011","11011","11011","11100","11101","11110","11111","1111
0","11111","11111","11111"),

("11111","11110","11001","11001","11100","11101","11101","11101","11101","111
01","11011","11001","11011","11110","11111","11111","11111","11011","11001","
11011","11101","11101","11101","11101","11101","11101","11110","11110","1110
1","11110","11111","11111"),

("11110","11101","11101","11101","11110","11110","11110","11101","11101","111
01","11100","11100","11101","11111","11111","11111","11111","11101","11100","
11100","11101","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11111","11111"),

("11110","11101","11111","11111","11111","11110","11110","11110","11101","111
01","11101","11101","11110","11111","11111","11111","11111","11110","11101","
11101","11101","11110","11110","11110","11110","11110","11110","11110","1111
0","11110","11110","11111"),

("11111","11110","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11110","11110","11101","11111","11111","11111","11111","11110","
11110","11110","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11110","11101","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11110","11110","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

```
("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11110","11110","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111")

);



   constant snow_blue : snow_tile := (

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
```

11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11110","11101","11110","11110","11110","11110","11110","11110","11101","
11110","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11110","11101","11101","11101","11101","11101","11101","11101","11101","
11110","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
01","11110","11110","11110","11110","11110","11110","11110","11110","11110","
11110","11101","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11110","11110","111
10","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11110","11110","11110","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11110","11110","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11101","11110","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11110","11110","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11101","11110","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11110","11101","11101","11101","11101","111
01","11110","11110","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11110","11101","11101","11110","11110","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11110","11110","11110","11110","11110","11110","11110","111
10","11110","11110","11110","11111","11111","11111","11111","11111","11110","
11111","11111","11110","11110","11110","11110","11110","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11101","11101","11110","11110","11110","11110","11110","111
10","11110","11101","11110","11111","11111","11111","11111","11110","11101","
11101","11110","11110","11110","11110","11110","11110","11111","11111","1111
0","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11110","11110","111
10","11110","11110","11110","11111","11111","11111","11111","11110","11110","

```
11110","11110","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11110","111
10","11110","11110","11111","11111","11111","11111","11111","11111","11110","
11110","11110","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11110","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),

("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111"),
```

```
("11111","11111","11111","11111","11111","11111","11111","11111","11111","111
11","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11111","11111","11111","11111","1111
1","11111","11111","11111")

);




 -- Signals for the video controller

 signal Hcount : unsigned(9 downto 0);  -- Horizontal position (0-800)

 signal Vcount : unsigned(9 downto 0);  -- Vertical position (0-524)

 signal sprite_height :   unsigned(9 downto 0); -- sprite point

 signal sprite_vertical : unsigned(9 downto 0);-- sprite point

 signal guard_height :   unsigned(9 downto 0); -- guard point

 signal guard_vertical : unsigned(9 downto 0);-- guard point

 signal EndOfLine, EndOfField : std_logic;


 signal vga_hblank, vga_hsync,

   vga_vblank, vga_vsync : std_logic;  -- Sync. signals


signal test1: integer := 0;

signal test2: integer := 0;

signal test3: integer := 0;

signal test4: integer := 0;

signal test5: integer := 0;

signal test6: integer := 0;
```

```
signal test7: integer := 0;

signal test8: integer := 0;


signal tilecounter: integer := 0;                    -- move sprite

signal tileflag: integer := 0;                       -- move sprite

signal guardcounter: integer := 0;                   -- move guard

signal guardflag: integer := 0;                      -- move guard


signal sprite_mode : integer := 0;              --sprite direction

signal guard_mode : integer := 0;        --guard direction


 signal sprite_h_down, sprite_v_down, sprite_down : std_logic;  -- sprite area

 signal sprite_h_up,sprite_v_up, sprite_up : std_logic;

 signal sprite_h_left,sprite_v_left, sprite_left : std_logic;

 signal sprite_h_right,sprite_v_right, sprite_right : std_logic;

 signal sprite_h_dead, sprite_v_dead,sprite_dead :std_logic;

 signal sprite_move_flag: std_logic;


 signal guard_h_down, guard_v_down, guard_down : std_logic;  -- guard area

 signal guard_h_up,guard_v_up, guard_up : std_logic;

 signal guard_h_left,guard_v_left, guard_left : std_logic;

 signal guard_h_right,guard_v_right, guard_right : std_logic;

 signal guard_move_flag: std_logic;
```

```
  signal map_h, map_v : std_logic;

  signal map_ice_change_h,map_ice_change_v : integer:=0;--indicate the position of
ice that need to change

  signal map_ice_change_flag: std_logic; -- map_ice change flag

  signal map_h_counter : integer := 0;

  signal map_v_counter : integer := 0;



  signal map_grape_change_h,map_grape_change_v : integer:=0;--indicate the
position of ice that need to change

  signal map_grape_change_flag: std_logic; -- map_ice change flag




  signal map_banana_change_h,map_banana_change_v : integer:=0;--indicate the
position of ice that need to change

  signal map_banana_change_flag: std_logic; -- map_ice change flag



  signal key_counter : integer := 0;   --  key counter for motion

  signal guard_counter : integer := 0;    -- key counter for guard motion

  signal counter_g : unsigned (21 downto 0):="0000000000000000000000";

  signal counter_s : unsigned (21 downto 0):="0000000000000000000000";

  signal gcounter_max : unsigned (21 downto 0):= "0000000000000000000000";

  signal readx : unsigned (15 downto 0) := "0000000000000000";                    --
for readdata
```

```
 signal clk1: std_logic := '0';                -- clk divide


 signal startflag : unsigned (1 downto 0) := "00";


--  signal mapping : map_level:= (

--
(("001","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("001","001","001","001","001","001","001","001","001","000","000","001","001","
001","001","001","001","001","001","001"),

--
("001","000","000","011","011","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("001","000","000","000","000","000","000","000","000","000","001","000","000","
000","001","001","001","001","111","000"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","111"),

--
("000","000","000","000","000","001","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","010","010","000","010","010","
000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),
```

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","111","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","111","111","111","111","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","011","011","011","011","000","000","000","001","000","000","000","000"),

--
("111","100","100","100","100","111","111","111","010","100","100","100","111","111","111","011","100","100","100","111")),

--

--
(("001","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("001","001","001","001","001","001","001","001","001","000","000","001","001","001","001","001","001","001","001","001"),

--
("001","000","000","000","010","010","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("001","000","000","111","111","111","001","001","000","000","000","000","000","000","000","011","011","000","000"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("001","000","011","000","000","000","000","000","000","000","000","000","000","000","000","011","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","011","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","010","001","001","001","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","010","001","001","001","000","000"),

--
("000","000","000","000","010","010","000","000","000","000","000","000","000","000","001","001","001","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","001","001","001","000","000"),

--
("111","100","100","100","100","111","111","111","010","100","100","100","111","111","111","011","100","100","100","111")),

--

--
(("001","000"),

--
("001","001","001","001","001","001","001","001","001","000","000","001","001","001","001","001","001","001","001"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("001","000","000","011","000","000","010","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","010","010","010","010","000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","111","000","000","000","000","000","000","000","001","000","000","000"),

--
("000","000","000","000","000","000","111","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","111","111","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","011","011","011","011","000","
000","000","001","000","000","000","000"),

--
("111","100","100","100","100","111","111","111","010","100","100","100","111","
111","111","011","100","100","100","111"))

--);--ice=01,banana=10,grape=11

--signal mapping : map_position:=

--
(("001","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("001","001","001","001","001","001","001","001","001","000","000","001","001","
001","001","001","001","001","001","001"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("001","000","000","000","000","000","000","000","000","000","111","000","000","
000","111","000","000","000","111","000"),

--
("001","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("000","000","000","000","000","111","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","010","010","010","010","000","
000","000","000","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

```
--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","111","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","000","000","000","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","111","111","111","111","000","000","000","001","000","000","000","000"),

--
("000","000","000","000","000","000","000","000","011","011","011","011","000","000","000","001","000","000","000","000"),

--
("100","100","100","100","001","010","100","100","100","001","011","100","100","100","000","000","000","000","000","000"));

signal mapping : map_level:= (

(("001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001"),

("001","010","000","010","000","010","000","010","001","001","111","111","001","001","001","001","001","001","001"),

("001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","011","001","001"),

("001","001","001","001","001","011","001","001","001","001","001","001","001","001","001","001","000","001","001"),

("001","001","001","001","011","001","001","001","001","001","001","001","001","001","001","001","011","001","001"),

("001","001","001","011","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001"),

("001","001","011","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001"),
```

("001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","011","001","001"),

("001","001","001","001","001","010","000","010","000","010","000","010","001","001","000","001","001","000","001","001"),

("001","001","001","001","001","001","001","001","001","001","001","001","001","001","000","001","001","011","001","001"),

("001","001","111","001","001","001","001","001","001","001","001","001","001","001","000","001","001","001","001","001"),

("001","001","001","111","001","001","001","001","001","001","001","001","001","001","000","001","001","001","001","001"),

("000","000","001","001","111","001","001","001","001","001","001","001","001","001","000","001","001","001","001","001"),

("000","000","001","001","001","001","001","001","001","001","001","001","001","001","000","001","001","001","001","001"),

("111","100","100","100","100","111","111","111","100","100","100","100","111","111","111","100","100","100","100","111")),

((("001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001"),

("001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001","001"),

("001","001","111","111","000","000","011","001","011","000","000","011","001","011","000","000","111","111","001","001"),

("001","001","111","000","000","000","000","010","000","000","000","000","010","000","000","000","000","111","001","001"),

("001","001","000","000","000","000","011","001","011","000","000","011","001","011","000","000","000","000","001","001"),

("001","001","000","000","000","000","000","010","000","000","000","000","010","000","000","000","000","000","001","001"),

("001","001","000","000","000","000","011","001","011","000","000","011","001","011","000","000","000","000","001","001"),

("001","001","000","000","000","000","000","010","000","000","000","000","010","
000","000","000","000","000","001","001"),

("001","001","000","000","000","000","011","001","011","000","000","011","001","
011","000","000","000","000","001","001"),

("001","001","000","000","000","000","000","010","000","000","000","000","010","
000","000","000","000","000","001","001"),

("001","001","111","000","000","000","011","001","011","000","000","011","001","
011","000","000","000","111","001","001"),

("001","001","111","111","000","000","000","010","000","000","000","000","010","
000","000","000","111","111","001","001"),

("000","000","001","001","001","001","001","001","001","001","001","001","001","
001","000","001","001","001","001","001"),

("000","000","001","001","001","001","001","001","001","001","001","001","001","
001","000","001","001","001","001","001"),

("111","100","100","100","100","111","111","111","100","100","100","100","111","
111","111","100","100","100","100","111")),

((("000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

("001","001","001","001","001","001","001","001","001","001","001","001","001","
001","001","001","001","001","001","001"),

("001","001","010","010","010","001","001","001","010","010","010","010","001","
001","001","010","010","010","001","001"),

("001","001","001","001","001","001","001","001","001","001","001","001","001","
001","001","001","001","001","001","001"),

("000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

("000","000","000","000","000","000","000","000","000","000","000","000","000","
000","000","000","000","000","000","000"),

("111","111","111","000","000","000","000","000","000","001","001","001","001","
001","000","001","001","001","001","001"),

("000","000","011","011","000","000","000","000","000","001","001","001","001","001","000","001","001","001","001","001"),

("000","000","000","000","000","000","000","000","000","001","001","011","001","001","000","001","001","011","001","001"),

("000","000","000","010","000","111","111","111","111","001","001","011","001","001","000","001","001","011","001","001"),

("000","000","000","010","000","000","111","111","000","001","001","011","001","001","000","001","001","011","001","001"),

("000","000","000","000","000","000","000","000","000","001","001","011","001","001","000","001","001","011","001","001"),

("000","000","000","000","000","000","000","000","000","001","001","001","001","001","000","001","001","001","001","001"),

("000","000","000","000","000","000","000","000","000","001","001","001","001","001","000","001","001","001","001","001"),

("111","100","100","100","100","111","111","111","100","100","100","100","111","111","111","100","100","100","100","111"))

);

signal ice_cream_R : std_logic_vector (4 downto 0);

signal ice_cream_G : std_logic_vector (4 downto 0);

signal ice_cream_B : std_logic_vector (4 downto 0);

signal image_address : unsigned(4 downto 0);

signal guard_R : std_logic_vector (4 downto 0);

signal guard_G : std_logic_vector (4 downto 0);

signal guard_B : std_logic_vector (4 downto 0);

signal guard_image_address : unsigned(4 downto 0);

```
----------signals for game info----------

signal num_address : integer;


signal num_R : std_logic_vector (4 downto 0);

signal num_G : std_logic_vector (4 downto 0);

signal num_B : std_logic_vector (4 downto 0);


signal gamelevel: integer := 0;

signal grape_record : integer := 0;

signal grape_recordten : integer :=0;

signal banana_record : integer := 0;

signal banana_recordten : integer := 0;


----------reset----------

signal  reset : integer :=  0;

signal  reset_flag : integer := 0;


----------


begin


sprite_rgb: ice_cream_controller port map
(clk,image_address,test1,test2,ice_cream_R,ice_cream_G,ice_cream_B);

guard_rgb : guard_controller port map
(clk,guard_image_address,test3,test4,guard_R,guard_G,guard_B);
```

```
info_control: info_controller port
map(clk,num_address,test5,test6,num_R,num_G,num_B);



 process (clk)

  begin

    if rising_edge(clk) then

      clk1 <= not clk1;

    end if;

  end process;

----------------------------------------------for read data-----------------------

process (clk)

begin

  if rising_edge(clk) then

     if reset_n = '0' then

       readdata  <= (others => '0');

     else

     if chipselect = '1' then

       if address = "00001" then

          if read = '1' then

         readdata <= readx;

       end if;

     end if;

   end if;

end if;
```

```
end if;
end process;



----------Ice_cream_direction----------
  process (clk)
  begin
    if rising_edge(clk) then

      if banana_record =10 then
            banana_recordten <= banana_recordten + 1;
            banana_record <= 0;
      end if;
          if grape_record =10 then
            grape_recordten <= grape_recordten + 1;
            grape_record <= 0;
          end if;

      if startflag = "00" then
                              guardflag <= 0;
                    end if;

      if reset_n = '0' then
        readx  <= (others => '0');
```

```
    else
      if tilecounter = 32 then                    -- see if tilecounter is 0
        tileflag <= 0;
----------------------------------------------------for sprite to move again----------
        readx <= "0000000000001111";                                    --15 for stop
moving
-------------------------------------------------------------------------------
       end if;
      if guardcounter = 32 then                    -- see if guardcounter is 0
        guardflag <= 0;
       end if;

      if chipselect = '1' then
-------------------------------------------reset readdata for sprite---------------------------------
       if  address= "10010" then                                                --18
           if write = '1' then
             readx <= "0000000000000000";
          end if;
       end if;
-----------------------------------------for select level-----------------------------------
       if  address= "01111" then                                                --15
          gamelevel <= gamelevel + 1;
          reset <= 1;
          if gamelevel = 3 then
          gamelevel <= 0;
```

```
            end if;
          end if;
-----------------------------------------for select grape record-----------------------------
        if address = "10000" then                                                    --16
          if write = '1' then
          grape_record <= grape_record + 1;
--          if grape_record =10 then
--            grape_recordten <= grape_recordten + 1;
--            grape_record <= 0;
--          end if;
          end if;
        end if;
-----------------------------------------for select banana record-----------------------------
        if address = "10001" then                                                    --17
          if write= '1' then
           banana_record <= banana_record + 1;
--          if banana_record =10 then
--            banana_recordten <= banana_recordten + 1;
--            banana_record <= 0;
--          end if;
          end if;
        end if;
-----------------------------------------------------for select ice cream----------------------
```

```
        if address = "00001"  then                      -- down 1
          if write = '1' then
            if sprite_mode /= 0 then
              sprite_mode <= 0;
            end if;

            if sprite_vertical /= writedata(9 downto 0)  then
                                    tileflag <= 1;                          --for move
sprite in vhdl
                                    end if;

          end if;
        elsif  address = "00000" then                   -- left 0
          if write = '1' then
                                     if sprite_mode /= 2 then
            sprite_mode <= 2;
          end if;

          if sprite_height /= writedata(9 downto 0)  then
                                    tileflag <= 1;                          --for move
sprite in vhdl
                                    end if;
                                   end if;

          elsif address = "00011"  then                 -- up 3
```

```
        if write = '1' then
                if sprite_mode /= 1 then
                sprite_mode <= 1;
                 end if;

              if sprite_vertical /= writedata(9 downto 0)  then
                                        tileflag <= 1;                          --for move
sprite in vhdl
                                        end if;



            end if;
        elsif address= "00010" then    -- right 2
           if write = '1' then
                if sprite_mode /= 3 then
                 sprite_mode <= 3;
               end if;
             if sprite_height /= writedata(9 downto 0) then
                                        tileflag <= 1;                          --for move
sprite in vhdl
                                        end if;

            end if;
           end if;
```

```
----------------------------------for select guard-------------------------------------------

        if address = "01001"  then                          -- down 9
        if write = '1' then


           if guard_vertical = writedata(9 downto 0) then
              guardflag <= 0;
             else
                                          guardflag <= 1;                          --for move  in vhdl
                                         end if;
             guard_mode <= 0;
           end if;
        elsif  address = "01000" then -- left 8
              if write = '1' then
             if guard_height = writedata(9 downto 0) then
              guardflag <= 0;
             else
                                          guardflag <= 1;                          --for move in vhdl
                                         end if;
             guard_mode <= 2;
           end if;
              elsif address = "01011"  then                  -- up 11
        if write = '1' then
```

```
                                    if guard_vertical = writedata(9 downto 0) then
            guardflag <= 0;
            else
                                     guardflag <= 1;                    --for move sprite
in vhdl
                                    end if;
            guard_mode <= 1;
          end if;
       elsif address= "01010" then    -- right 10
           if write = '1' then


                                if guard_height = writedata(9 downto 0) then
            guardflag <= 0;
            else
                                     guardflag <= 1;                    --for move sprite
in vhdl
                                    end if;
            guard_mode <= 3;
          end if;
         end if;


     end if;
```

```
        if address = "01110"  then
          if write = '1' then
             sprite_mode <= 4;                              -- dead 14
           end if;
        end if;

        if address = "11110" then
                        if write = '1' then
             gcounter_max <= "00001111111111111111";                              --
30  set speed
           end if;
        end if;

       if address = "11111" then
                        if write = '1' then
             gcounter_max <= "00001111111111111111";                              --
31
           end if;
        end if;

       if address = "11101"  then
                --29  reset
         if write = '1' then
            reset <= 1;
         end if;
       end if;
```

```
        if reset_flag = 1 then
            reset <= 0;
         end if;

        end if;
       end if;

  end process;
-----------------------------------move sprite in vhdl---------------------------------
process(clk1)
begin

if rising_edge(clk1) then

   if startflag = "00" then                                          --initialize
   sprite_vertical <= "0000100000";
   sprite_height <= "0000000000";
   guard_vertical <= "0000100000";
   guard_height <= "0111000000";
   startflag <= "01";
   end if;

    if reset = 1 then                  -- need to reset
```

```
    startflag <= "00";
    counter_g <= "00000000000000000000";
    reset_flag <= 1;
  else
    reset_flag <= 0;
  end if;

        if counter_s = "00001111111111111111" then
   if tileflag = 1 then
    if sprite_mode = 0 then
    sprite_vertical <= sprite_vertical + 1;
    elsif sprite_mode = 1 then
    sprite_vertical <= sprite_vertical - 1 ;
    elsif sprite_mode = 2 then
    sprite_height <= sprite_height - 1;
    elsif sprite_mode = 3 then
    sprite_height <= sprite_height + 1;
    end if;
    tilecounter <= tilecounter + 1;                    --every time counter ++ to 32 for
one tile
    key_counter <= key_counter + 1;
     if key_counter = 15 then
     key_counter <= 0;
    end if;
   else
```

```
    tilecounter <= 0;                                   --reset tilecounter
   end if;
  counter_s<="00000000000000000000";
 else counter_s <= counter_s + 1;
 end if;
  if counter_g = gcounter_max then
   if guardflag = 1 then
    if guard_mode = 0 then
    guard_vertical <= guard_vertical + 1;
    elsif guard_mode = 1 then
    guard_vertical <= guard_vertical - 1 ;
    elsif guard_mode = 2 then
    guard_height <= guard_height - 1;
    elsif guard_mode = 3 then
    guard_height <= guard_height + 1;
    end if;
    guardcounter <= guardcounter + 1;                   --every time counter ++
to 32 for one tile
    guard_counter <= guard_counter + 1;
     if guard_counter = 15 then
     guard_counter <= 0;
    end if;
   else
   guardcounter <= 0;                                   --reset tilecounter
  end if;
```

```
   counter_g<="00000000000000000000";
  else counter_g <= counter_g + 1;
   end if;
end if;


end process;



----------map_ice_banana_grape_change--------generate or eliminate ice wall--------
process (clk)
begin
if rising_edge(clk) then
     if chipselect = '1' then
        if address = "00100"  then                          -- vertical 4
          if write = '1' then
            map_ice_change_v <=to_integer(writedata(9 downto 0));
          end if;
        elsif address = "00101" then             -- horizontal 5
          if write='1' then
            map_ice_change_h <=to_integer(writedata(9 downto 0));
            map_ice_change_flag <= '1';
          end if;
        elsif address = "00110"  then                       -- vertical address : 6
          if write = '1' then
```

```
            map_grape_change_v <=to_integer(writedata(9 downto 0));
          end if;
        elsif address = "00111" then            -- horizontal address : 7
          if write='1' then
            map_grape_change_h <=to_integer(writedata(9 downto 0));
            map_grape_change_flag <= '1';
          end if;
        elsif address = "01100"  then                    -- vertical address: 12
          if write = '1' then
            map_banana_change_v <=to_integer(writedata(9 downto 0));
          end if;
        elsif address = "01101" then            -- horizontal address : 13
          if write='1' then
            map_banana_change_h <=to_integer(writedata(9 downto 0));
            map_banana_change_flag <= '1';
          end if;
        end if;
    elsif map_ice_change_flag='1' then
       case mapping(gamelevel)(map_ice_change_v)(19-map_ice_change_h) is
        when "001" => mapping(gamelevel)(map_ice_change_v)(19-
map_ice_change_h)<="000";
        when "000" => mapping(gamelevel)(map_ice_change_v)(19-
map_ice_change_h)<="001";
        when others=> null;
       end case;
```

```
        map_ice_change_flag <= '0';

     elsif map_grape_change_flag='1' then

        case mapping(gamelevel)(map_grape_change_v)(19-map_grape_change_h) is

         when "011" => mapping(gamelevel)(map_grape_change_v)(19-
map_grape_change_h)<="000";

         when "000" => mapping(gamelevel)(map_grape_change_v)(19-
map_grape_change_h)<="011";

         when others=>null;

        end case;

        map_grape_change_flag <= '0';

     elsif map_banana_change_flag='1' then

        case mapping(gamelevel)(map_banana_change_v)(19-map_banana_change_h)
is

         when "010" => mapping(gamelevel)(map_banana_change_v)(19-
map_banana_change_h)<="000";

         when "000" => mapping(gamelevel)(map_banana_change_v)(19-
map_banana_change_h)<="010";

         when others =>null;

        end case;

        map_banana_change_flag <= '0';

     end if;

end if;

end process;



--------------------------------------------------------------------------------------------
```

```
HCounter : process (clk1)
 begin
  if rising_edge(clk1) then
    if reset_n = '0' then
      Hcount <= (others => '0');
    elsif EndOfLine = '1' then
      Hcount <= (others => '0');
    else
      Hcount <= Hcount + 1;
    end if;
  end if;
 end process HCounter;

 EndOfLine <= '1' when Hcount = HTOTAL - 1 else '0';

 VCounter: process (clk1)
 begin
  if rising_edge(clk1) then
    if reset_n = '0' then
      Vcount <= (others => '0');
    elsif EndOfLine = '1' then
      if EndOfField = '1' then
        Vcount <= (others => '0');
```

```
      else
        Vcount <= Vcount + 1;
      end if;
    end if;
  end if;
end process VCounter;

EndOfField <= '1' when Vcount = VTOTAL - 1 else '0';

-- State machines to generate HSYNC, VSYNC, HBLANK, and VBLANK

HSyncGen : process (clk1)
begin
  if rising_edge(clk1) then
    if reset_n = '0' or EndOfLine = '1' then
      vga_hsync <= '1';
    elsif Hcount = HSYNC - 1 then
      vga_hsync <= '0';
    end if;
  end if;
end process HSyncGen;

HBlankGen : process (clk1)
begin
```

```
  if rising_edge(clk1) then
    if reset_n = '0' then
      vga_hblank <= '1';
    elsif Hcount = HSYNC + HBACK_PORCH  then
      vga_hblank <= '0';
    elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE  then
      vga_hblank <= '1';
    end if;
  end if;
end process HBlankGen;

VSyncGen : process (clk1)
begin
  if rising_edge(clk1) then
    if reset_n = '0' then
      vga_vsync <= '1';
    elsif EndOfLine ='1' then
      if EndOfField = '1' then
        vga_vsync <= '1';
      elsif Vcount = VSYNC - 1 then
        vga_vsync <= '0';
      end if;
    end if;
  end if;
```

```
  end process VSyncGen;


  VBlankGen : process (clk1)
  begin
    if rising_edge(clk1) then
      if reset_n = '0' then
        vga_vblank <= '1';
      elsif EndOfLine = '1' then
        if Vcount = VSYNC + VBACK_PORCH - 1 then
          vga_vblank <= '0';
        elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
          vga_vblank <= '1';
        end if;
      end if;
    end if;
  end process VBlankGen;


  -- map everything
 MapGen : process (clk1)
  begin
    if rising_edge(clk1) then
      if reset_n = '0' or Hcount = HSYNC + HBACK_PORCH + 32*map_h_counter  then
    map_h_counter <= map_h_counter + 1;
      if map_h_counter = 20 then  -- map_h_counter= 20
```

```
   map_h_counter <= 0;
  end if;
 end if;
  if  reset_n = '0' or Vcount = VSYNC + VBACK_PORCH + 32*map_v_counter  then

          map_v_counter <= map_v_counter + 1;
        if map_v_counter = 15 then  -- 15
          map_v_counter <= 0;
        end if;
     end if;
   end if;
 end process MapGen;




-- sprite generator


  RectangleHGen : process (clk1)
  begin
    if rising_edge(clk1) then
      if  reset_n = '0' or Hcount = HSYNC + HBACK_PORCH +  sprite_height  then
        if sprite_mode = 0 then
        sprite_h_down <= '1';
        sprite_h_up <= '0';
        sprite_h_left <='0';
```

```
sprite_h_right <='0';
sprite_h_dead <= '0';
elsif sprite_mode = 1 then
sprite_h_up <= '1';
sprite_h_down <= '0';
sprite_h_left <='0';
sprite_h_right <='0';
sprite_h_dead <= '0';
elsif sprite_mode = 2 then
sprite_h_left <= '1';
sprite_h_up <= '0';
sprite_h_down <='0';
sprite_h_right <='0';
sprite_h_dead <= '0';
elsif sprite_mode = 3 then
sprite_h_right <= '1';
sprite_h_up <= '0';
sprite_h_left <='0';
sprite_h_down <='0';
sprite_h_dead <= '0';
elsif sprite_mode = 4 then
sprite_h_dead <= '1';
sprite_h_down <= '0';
sprite_h_up <= '0';
```

```
      sprite_h_left <= '0';
      sprite_h_right <= '0';
     end if;
    end if;
    if reset_n = '0' or Hcount = HSYNC + HBACK_PORCH + sprite_height+32 then
      sprite_h_down <= '0';
      sprite_h_up <= '0';
      sprite_h_left <= '0';
      sprite_h_right <= '0';
      sprite_h_dead <= '0';
    end if;



  end if;
end process RectangleHGen;

RectangleVGen : process (clk1)
begin
  if rising_edge(clk1) then
    if reset_n = '0' then
      sprite_v_up <= '0';
      sprite_v_down <= '0';
      sprite_v_left <= '0';
      sprite_v_right <= '0';
```

```
  sprite_v_dead <= '0';
elsif EndOfLine = '1' then

  if Vcount >= VSYNC + VBACK_PORCH + sprite_vertical -1  then
  if sprite_mode = 0 then
  sprite_v_down <= '1';
  sprite_v_up <= '0';
  sprite_v_left <='0';
  sprite_v_right <='0';
  sprite_v_dead <= '0';
  elsif sprite_mode = 1 then
  sprite_v_up <= '1';
  sprite_v_down <= '0';
  sprite_v_left <='0';
  sprite_v_right <='0';
  sprite_v_dead <= '0';
  elsif sprite_mode = 2 then
  sprite_v_left <= '1';
  sprite_v_up <= '0';
  sprite_v_down <='0';
  sprite_v_right <='0';
  sprite_v_dead <= '0';
  elsif sprite_mode = 3 then
  sprite_v_right <= '1';
```

```
      sprite_v_up <= '0';
      sprite_v_left <='0';
      sprite_v_down <='0';
      sprite_v_dead <= '0';
      elsif sprite_mode = 4 then
        sprite_v_dead <= '1';
        sprite_v_down <= '0';
        sprite_v_up <= '0';
        sprite_v_left <= '0';
        sprite_v_right <= '0';
     end if;
     end if;
   if Vcount >= VSYNC + VBACK_PORCH  + sprite_vertical + 31 then
        sprite_v_down <= '0';
        sprite_v_up <= '0';
        sprite_v_left <= '0';
        sprite_v_right <= '0';
        sprite_v_dead <= '0';
                    end if;
    end if;
 end if;
end process RectangleVGen;


sprite_up <= sprite_h_up and sprite_v_up;
```

```
  sprite_down <= sprite_h_down and sprite_v_down;

  sprite_left <= sprite_h_left and sprite_v_left;

  sprite_right <= sprite_h_right and sprite_v_right;

  sprite_dead <= sprite_h_dead and sprite_v_dead;

  sprite_move_flag<= sprite_up or sprite_down or sprite_left or sprite_right or
sprite_dead;

-- guard generator


  GuardHGen : process (clk1)

  begin

    if rising_edge(clk1) then

      if  reset_n = '0' or Hcount = HSYNC + HBACK_PORCH +  guard_height  then

        if guard_mode = 0 then

        guard_h_down <= '1';

        guard_h_up <= '0';

        guard_h_left <='0';

        guard_h_right <='0';

        elsif guard_mode = 1 then

        guard_h_up <= '1';

        guard_h_down <= '0';

        guard_h_left <='0';

        guard_h_right <='0';

        elsif guard_mode = 2 then

        guard_h_left <= '1';

        guard_h_up <= '0';
```

```
      guard_h_down <='0';
      guard_h_right <='0';
      elsif guard_mode = 3 then
      guard_h_right <= '1';
      guard_h_up <= '0';
      guard_h_left <='0';
      guard_h_down <='0';
     end if;
    end if;
    if reset_n = '0' or Hcount = HSYNC + HBACK_PORCH + guard_height+32 then
      guard_h_down <= '0';
      guard_h_up <= '0';
      guard_h_left <= '0';
      guard_h_right <= '0';
    end if;


  end if;
end process GuardHGen;

GuardVGen : process (clk1)
begin
  if rising_edge(clk1) then
    if reset_n = '0' then
```

```
  guard_v_up <= '0';
  guard_v_down <= '0';
  guard_v_left <= '0';
  guard_v_right <= '0';
elsif EndOfLine = '1' then

  if Vcount >= VSYNC + VBACK_PORCH + guard_vertical -1  then
  if guard_mode = 0 then
  guard_v_down <= '1';
  guard_v_up <= '0';
  guard_v_left <='0';
  guard_v_right <='0';
  elsif guard_mode = 1 then
  guard_v_up <= '1';
  guard_v_down <= '0';
  guard_v_left <='0';
  guard_v_right <='0';
  elsif guard_mode = 2 then
  guard_v_left <= '1';
  guard_v_up <= '0';
  guard_v_down <='0';
  guard_v_right <='0';
  elsif guard_mode = 3 then
  guard_v_right <= '1';
```

```
      guard_v_up <= '0';
      guard_v_left <='0';
      guard_v_down <='0';
     end if;
     end if;
   if Vcount >= VSYNC + VBACK_PORCH  + guard_vertical + 31 then
       guard_v_down <= '0';
       guard_v_up <= '0';
       guard_v_left <= '0';
       guard_v_right <= '0';
     end if;
    end if;
 end if;
end process GuardVGen;

guard_up <= guard_h_up and guard_v_up;
guard_down <= guard_h_down and guard_v_down;
guard_left <= guard_h_left and guard_v_left;
guard_right <= guard_h_right and guard_v_right;
guard_move_flag<= guard_up or guard_down or guard_left or guard_right;

-- Registered video signals going to the video DAC
```

```
 VideoOut: process (clk1, reset_n)
 begin
  if reset_n = '0' then
    VGA_R <= "0000000000";                    --0100000000
    VGA_G <= "0000000000";          --0110000000
    VGA_B <= "0000000000";          --0100000000
  elsif clk1'event and clk1= '1' then
    ------------------------------------------------------------map ice----------------------------------
    if map_v_counter /= 0 and map_h_counter /=0 then
      if mapping(gamelevel)(map_v_counter-1)(20 - map_h_counter) /= "000" then
      VGA_R(1 downto 0) <= "11";
      VGA_G(1 downto 0) <= "11";
      VGA_B(1 downto 0) <= "11";

       test5 <= to_integer(Vcount - (VSYNC + VBACK_PORCH )-32*(map_v_counter-1));
       test6 <= to_integer(Hcount - (HSYNC + HBACK_PORCH )-32*(map_h_counter-
1));

     case  mapping(gamelevel)(map_v_counter-1)(20 - map_h_counter) is
       when "001" => VGA_R(9 downto 2) <=ice_red(test5)(test6);
                     VGA_G(9 downto 2) <=ice_green(test5)(test6);
                     VGA_B(9 downto 2) <=ice_blue(test5)(test6);
       when "010" => VGA_R(9 downto 2) <=banana_red(test5)(test6);
                     VGA_G(9 downto 2) <=banana_green(test5)(test6);
                     VGA_B(9 downto 2) <=banana_blue(test5)(test6);
```

```
        when "011" => VGA_R(9 downto 2) <=grape_red(test5)(test6);
                      VGA_G(9 downto 2) <=grape_green(test5)(test6);
                      VGA_B(9 downto 2) <=grape_blue(test5)(test6);
        when "111" => VGA_R(9 downto 5) <= snow_red(test5)(test6);
                      VGA_R(4 downto 0) <= "11111";
                      VGA_G(9 downto 5) <= snow_green(test5)(test6);
                      VGA_G(4 downto 0) <= "11111";
                                          VGA_B(9 downto 5) <=
snow_blue(test5)(test6);
                                          VGA_B(4 downto 0) <= "11111";
        when "100"=>
                      case map_h_counter is
                      when 2 => num_address <= 11; -- LLL
                      when 3 => num_address <= 12; -- VVV
                      when 4 => num_address <= 10; -- xxx
                      when 5 => num_address <= gamelevel;

                      when 10 => num_address <= 10;
                      when 11 => num_address <= banana_recordten;
                      when 12 => num_address <= banana_record;

                      when 17 => num_address <= 10;
                      when 18 => num_address <=  grape_recordten;
                      when 19 => num_address <= grape_record;
```

```
--                  when 18 => num_address <= 10;
--                  when 19 => num_address <= 10;
                    when others=> null;
                    end case;
                    VGA_R(4 downto 2) <="111";
                    VGA_G(4 downto 2) <="111";
                    VGA_B(4 downto 2) <="111";
                    VGA_R(9 downto 5) <=num_R(4 downto 0);
                    VGA_G(9 downto 5) <=num_G(4 downto 0);
                    VGA_B(9 downto 5) <=num_B(4 downto 0);

       when others=>null;
     end case;


   elsif vga_hblank = '0' and vga_vblank ='0' then
     VGA_R <= "1111111111";
     VGA_G <= "1111111111";
     VGA_B <= "1111111111";
   else
     VGA_R <= "0000000000";
     VGA_G <= "0000000000";
     VGA_B <= "0000000000";
   end if;
```

```
elsif vga_hblank = '0' and vga_vblank ='0' then
   VGA_R <= "1111111111";
   VGA_G <= "1111111111";
   VGA_B <= "1111111111";
 else
   VGA_R <= "0000000000";
   VGA_G <= "0000000000";
   VGA_B <= "0000000000";
 end if;
------------------------------------------------------------sprite ------------------------------------
if sprite_move_flag ='1' then
    test1 <=  to_integer(Vcount - sprite_vertical) -VSYNC - VBACK_PORCH;
    test2 <=  to_integer(Hcount - sprite_height) -HSYNC - HBACK_PORCH ;
if sprite_down = '1'  then
   if key_counter <4   then
    image_address <= "00000";
   elsif key_counter > 7 and key_counter < 12   then
    image_address <= "00000";
   elsif key_counter >3 and key_counter < 8 then
    image_address <= "00001";
   elsif key_counter > 11 then
    image_address <= "00010";
  end if;
  elsif sprite_up = '1' then
```

```
 if key_counter <4   then
  image_address <= "00011";
 elsif key_counter > 7 and key_counter < 12   then
  image_address <= "00011";
 elsif key_counter >3 and key_counter < 8 then
 image_address <= "00100";
 elsif key_counter > 11 then
  image_address <= "00101";
end if;
elsif sprite_left = '1' then
  if key_counter <4   then
  image_address <= "00110";
 elsif key_counter > 7 and key_counter < 12   then
  image_address <= "00110";
 elsif key_counter >3 and key_counter < 8 then
  image_address <= "00111";
 elsif key_counter > 11 then
  image_address <= "01000";
end if;
elsif sprite_right = '1' then

  if key_counter <4   then
 image_address <= "01001";
 elsif key_counter > 7 and key_counter < 12   then
```

```
   image_address <= "01001";
   elsif key_counter >3 and key_counter < 8 then
   image_address <= "01010";
   elsif key_counter > 11 then
   image_address <= "01011";
  end if;

  elsif sprite_dead = '1' then
   image_address <= "01100";
 end if;
 if ice_cream_R /= "00000" then
 VGA_R(9 downto 5) <= ice_cream_R(4 downto 0);
  VGA_R(4 downto 0) <= "11111";
 end if;
  if ice_cream_G /= "00000" then
 VGA_G(9 downto 5) <= ice_cream_G(4 downto 0);
 VGA_G(4 downto 0) <= "11111";
 end if;
   if ice_cream_B /= "00000" then
 VGA_B(9 downto 5) <= ice_cream_B(4 downto 0);
 VGA_B(4 downto 0) <= "11111";
 end if;
end if;
```

```
-----------------------------------------------------------guard ------------------------------------
 if guard_move_flag ='1' then
     test3 <=  to_integer(Vcount - guard_vertical) -VSYNC - VBACK_PORCH;
     test4 <=  to_integer(Hcount - guard_height) -HSYNC - HBACK_PORCH;
 if guard_down = '1'  then
    if guard_counter <4   then
    guard_image_address <= "00000";
    elsif guard_counter > 7 and guard_counter < 12   then
    guard_image_address <= "00000";
    elsif guard_counter >3 and guard_counter < 8 then
     guard_image_address <= "00001";
    elsif guard_counter > 11 then
     guard_image_address <= "00010";
   end if;
   elsif guard_up = '1' then
    if guard_counter <4   then
    guard_image_address <= "00011";
    elsif guard_counter > 7 and guard_counter < 12   then
     guard_image_address <= "00011";
    elsif guard_counter >3 and guard_counter < 8 then
     guard_image_address <= "00100";
    elsif guard_counter > 11 then
     guard_image_address <= "00101";
```

```
 end if;
 elsif guard_left = '1' then
  if guard_counter < 4   then
  guard_image_address <= "00110";
  elsif guard_counter > 7 and guard_counter < 12   then
      guard_image_address <= "00110";
  elsif guard_counter >3 and guard_counter < 8 then
        guard_image_address <= "00111";
  elsif guard_counter > 11 then
        guard_image_address <= "01000";
 end if;
 elsif guard_right = '1' then
  if guard_counter <4   then
   guard_image_address <= "01001";
  elsif guard_counter > 7 and guard_counter < 12   then
   guard_image_address <= "01001";
  elsif guard_counter >3 and guard_counter < 8 then
   guard_image_address <= "01010";
  elsif guard_counter > 11 then
   guard_image_address <= "01011";
 end if;
end if;
  if guard_R /= "11111" then
VGA_R(9 downto 5) <= guard_R(4 downto 0);
```

```
    VGA_R(4 downto 0) <= "11111";
   end if;
    if guard_G /= "11111" then
   VGA_G(9 downto 5) <= guard_G(4 downto 0);
   VGA_G(4 downto 0) <= "11111";
   end if;
     if guard_B /= "11111" then
   VGA_B(9 downto 5) <= guard_B(4 downto 0);
   VGA_B(4 downto 0) <= "11111";
   end if;
 end if;




 end if;
end process VideoOut;


VGA_CLK <= clk1;
VGA_HS <= not vga_hsync;
VGA_VS <= not vga_vsync;
VGA_SYNC <= '0';
VGA_BLANK <= not (vga_hsync or vga_vsync);
```

```
end rtl;
```

irComp.vhd

```
-- Haodan Huang
--
-- This component is basically a 26-bit counter
--
-- When the counter is 1, irq will be '1'
-- The irq is reset to '0' by the cpu, when the cpu wants to write to this
-- component, the irq will become '0'
--
-- The address, readdata, and writedata are effectively ignored

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity irComp is port (
  clk, reset_n, chipselect, read, write, address : in std_logic;
  readdata  : out std_logic_vector(15 downto 0);
  writedata : in  std_logic_vector(15 downto 0);
  irq       : out std_logic);
end irComp;
```

```
architecture rtl of irComp is
  signal data    : std_logic_vector(15 downto 0);
  signal counter : unsigned(25 downto 0);
  signal counter_max : unsigned(25 downto 0);
  signal flag : integer := 0;
  signal f_flag: integer := 0;
begin

  process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
        data <= (others => '0');
      else
        if chipselect = '1' then
          if address = '1' then
            if write = '1' then
              data <= writedata;
            elsif read = '1' then
              readdata <= data;
            end if;
          end if;
        end if;
      end if;
```

```
   end if;
  end process;


process (clk)
begin
  if rising_edge(clk) then
     if reset_n = '0' then
       counter_max <= (others => '0');
   else
     if data(1 downto 0) = "00" then
       counter_max <= "11111111111111111111111111";
     elsif data(1 downto 0) = "01" then
       counter_max <= "01111111111111111111111111";
     elsif data(1 downto 0) = "10" then
       counter_max <= "01111111111111111111111111";
     end if;
  end if;
 end if;
end process;


  process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
```

```
      counter <= (others => '0');
    else
      counter <= counter +1 ;
    end if;
    if counter = counter_max then
      counter <= (others => '0');
      flag <= 1;
    end if;
   if f_flag = 1 then
     flag <= 0;
   end if;
  end if;
end process;

process (clk)
begin
  if rising_edge(clk) then
    if reset_n = '0' then
      irq <= '0';
    else
       if flag = 0 then
        f_flag <= 0;
       end if;
      if flag = 1 then
```

```
        irq <= '1';
        f_flag <= 1;
       elsif write = '1' and chipselect = '1' then
        irq <= '0';  -- important to reset the irq
       end if;
     end if;
   end if;
 end process;

end rtl;
```

de2_wm8731.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_wm8731_audio is
port (
   clk : in std_logic;       --  Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
   reset_n : in std_logic;

   write      : in  std_logic;
```

```
        read      : in  std_logic;
    chipselect : in  std_logic;
    address    : in  unsigned(4 downto 0);
    writedata  : in  unsigned(15 downto 0);
    readdata   : out unsigned(15 downto 0);

    -- Audio interface signals
    AUD_ADCLRCK  : out  std_logic;   --    Audio CODEC ADC LR Clock
    AUD_ADCDAT   : in   std_logic;   --    Audio CODEC ADC Data
    AUD_DACLRCK  : out  std_logic;   --    Audio CODEC DAC LR Clock
    AUD_DACDAT   : out  std_logic;   --    Audio CODEC DAC Data
    AUD_BCLK     : inout std_logic  --    Audio CODEC Bit-Stream Clock
  );
end  de2_wm8731_audio;

architecture rtl of de2_wm8731_audio is

    signal lrck : std_logic;
    signal bclk : std_logic;
    signal xck  : std_logic;

    signal lrck_divider : unsigned(11 downto 0);
    signal bclk_divider : unsigned(3 downto 0);
```

```
    signal set_bclk : std_logic;
    signal set_lrck : std_logic;
    signal clr_bclk : std_logic;
    signal lrck_lat : std_logic;

    signal shift_out : unsigned(15 downto 0);

    signal sin_out     : unsigned(15 downto 0);
    signal sin_counter : unsigned(11 downto 0);

-----------
type rom_type is array(920 downto 0) of unsigned(15 downto 0);

    constant ROM1 : rom_type :=
(

  X"2c21",
    X"ba0e",
    X"9602",
    X"34a4",
    X"7347",
    X"f0b0",
    X"8621",
    X"f2b1",
```

X"75dd",

X"2889",

X"9364",

X"bcbe",

X"5b11",

X"5cb1",

X"c1a9",

X"8e13",

X"ff91",

X"74ab",

X"3488",

X"9b9e",

X"b06b",

X"520e",

X"61e3",

X"c495",

X"8fc2",

X"1f31",

X"7840",

X"054c",

X"8835",

X"e464",

X"6670",

X"486a",

X"b44d",

X"ab11",

X"4b4e",

X"62ba",

X"d430",

X"8ef7",

X"e630",

X"68a5",

X"43f3",

X"ab9e",

X"a6a3",

X"4062",

X"67a7",

X"d739",

X"8fcd",

X"0bf4",

X"73a3",

X"1967",

X"9aa3",

X"d308",

X"5cd5",

X"3edf",

X"bbd3",

X"a4e1",

X"fc3b",

X"5f68",

X"e964",

X"a3ce",

X"04b5",

X"52d9",

X"39c9",

X"cd35",

X"b59c",

X"23c5",

X"4d59",

X"f290",

X"b1ba",

X"fa1f",

X"49bc",

X"1942",

X"c1bf",

X"d912",

X"34aa",

X"3087",

X"dbae",

X"c8c2",

X"05f6",

X"3968",

X"163a",

X"d075",

X"de98",

X"2446",

X"2591",

X"e6c1",

X"d5ea",

X"0c4c",

X"2b0e",

X"fe78",

X"d9f1",

X"f638",

X"1fab",

X"107e",

X"e4b3",

X"e8e8",

X"167a",

X"1b23",

X"f384",

X"e414",

X"fe15",

X"1d6a",

X"104e",

X"ec99",

X"ede7",

X"0d85",

X"128d",

X"f916",

X"ed73",

X"01d7",

X"13d6",

X"01ef",

X"f1bc",

X"fc09",

X"0bf0",

X"097c",

X"f748",

X"f6ed",

X"041f",

X"0a34",

X"fc28",

X"fc7a",

X"f7e3",

X"0968",

X"efe6",

X"2db1",

X"fae9",

X"95ce",

X"9945",

X"d84f",

X"66b1",

X"6870",

X"17b0",

X"91e5",

X"a14c",

X"b5dd",

X"5b32",

X"6702",

X"3a7f",

X"98b1",

X"a4fe",

X"a259",

X"5396",

X"5b85",

X"6dbe",

X"dc36",

X"985c",

X"9c08",

X"32b1",

X"6443",

X"6bc2",

X"fc21",

X"9541",

X"9a18",

X"d48d",

X"65ac",

X"67f3",

X"1c82",

X"9161",

X"a3a4",

X"b13a",

X"610a",

X"5ad5",

X"64e8",

X"bfeb",

X"9a8c",

X"abed",

X"4bbd",

X"5a36",

X"68e2",

X"e246",

X"9bdd",

X"9ef1",

X"2cd0",

X"613e",

X"66d0",

X"fdfa",

X"98da",

X"9f1c",

X"d260",

X"607a",

X"63c6",

X"1f26",

X"96ae",

X"a72e",

X"b3d6",

X"5c70",

X"5745",

X"6172",

X"c364",

X"9db8",

X"adee",

X"4b08",

X"5a29",

X"693d",

X"e8ae",

X"a54a",

X"a25c",

X"f070",

X"5c71",

X"5bf6",

X"02d4",

X"a3ae",

X"ae62",

X"d64d",

X"4fca",

X"52f8",

X"1d83",

X"ac41",

X"ba02",

X"c2ff",

X"4893",

X"4172",

X"4a00",

X"d564",

X"b5a9",

X"c0ec",

X"328b",

X"4067",

X"4a64",

X"ee1f",

X"bdf5",

X"bbed",

X"f3e8",

X"427f",

X"3de7",

X"0327",

X"bf28",

X"c8ac",

X"dddb",

X"3b14",

X"3337",

X"32d8",

X"d92a",

X"cc99",

X"da57",

X"2dfa",

X"2d54",

X"3113",

X"e770",

X"cff0",

X"d6c6",

X"1d7f",

X"2857",

X"2d83",

X"f635",

X"d85a",

X"d7f5",

X"f7cf",

X"2847",

X"2395",

X"02c1",

X"dc99",

X"e0dd",

X"edd2",

X"1bf6",

X"1cca",

X"0bd9",

X"e198",

X"e64a",

X"e8d8",

X"1af3",

X"186d",

X"1c09",

X"f2ff",

X"e8c0",

X"e574",

X"0259",

X"1b45",

X"1ae7",

X"fbbd",

X"e949",

X"ebf4",

X"fa2a",

X"15ca",

X"11df",

X"018c",

X"ee53",

X"f0a3",

X"f604",

X"1088",

X"0ed2",

X"0e60",

X"f4e2",

X"f5b1",

X"f866",

X"0b10",

X"0b07",

X"0caa",

X"fbb8",

X"f599",

X"f658",

X"04e2",

X"078b",

X"094a",

X"fe17",

X"fba1",

X"f5e0",

X"11a6",

X"b980",

X"96f2",

X"37e1",

X"713c",

X"e88e",

X"869e",

X"f431",

X"76ca",

X"2dd1",

X"a4a2",

X"9a13",

X"2444",

X"78c3",

X"f40a",

X"87c1",

X"ee6f",

X"75c4",

X"2d7e",

X"97c8",

X"b2f7",

X"5372",

X"60f7",

X"c269",

X"9147",

X"242f",

X"779d",

X"fc47",

X"8934",

X"e454",

X"6a75",

X"4181",

X"b8ac",

X"9554",

X"1030",

X"75ec",

X"0710",

X"8e33",

X"dd8e",

X"6ab4",

X"3d5a",

X"a768",

X"a8bb",

X"41c3",

X"6700",

X"d507",

X"90df",

X"12f0",

X"73eb",

X"131d",

X"99c0",

X"d4cc",

X"5dda",

X"3d31",

X"babb",

X"a5db",

X"fc9b",

X"6259",

X"15bb",

X"a994",

X"d7a1",

X"4b9a",

X"3997",

X"c7f5",

X"b839",

X"23f9",

X"4db5",

X"eeff",

X"b2dc",

X"fde7",

X"4c9d",

X"137b",

X"c7b3",

X"c5ff",

X"1cb2",

X"3634",

X"d6e2",

X"cbfb",

X"071d",

X"39e6",

X"1545",

X"cf7d",

X"e01c",

X"25f9",

X"275f",

X"e87b",

X"d548",

X"0da8",

X"2a70",

X"fdd4",

X"d9a9",

X"f728",

X"204b",

X"10ca",

X"e852",

X"e2b3",

X"08e7",

X"2097",

X"0256",

X"e38f",

X"ffa0",

X"1d3c",

X"1016",

X"ec19",

X"ef02",

X"0ea6",

X"1215",

X"f8d2",

X"ed6a",

X"0246",

X"13c0",

X"0147",

X"f17b",

X"fd09",

X"0c05",

X"0a29",

X"f9c1",

X"f4ab",

X"ff9c",

X"09db",

X"0294",

X"f990",

X"fb69",

X"0078",

X"ff3f",

X"0136",

X"fdfc",

X"0338",

X"fdbb",

X"0e05",

X"259b",

X"9c25",

X"a192",

X"9f8d",

X"9f74",

X"340b",

X"6e96",

X"53ab",

X"728e",

X"fa1e",

X"8f83",

X"a96a",

X"9678",

X"b81a",

X"4f77",

X"65d4",

X"5b02",

X"66fe",

X"dcb0",

X"9176",

X"aaef",

X"913e",

X"d419",

X"6121",

X"453f",

X"519e",

X"8000",

X"eb1c",

X"7c1d",

X"9449",

X"9de6",

X"2165",

X"6771",

X"38fd",

X"6ced",

X"5139",

X"abba",

X"8412",

X"9046",

X"a591",

X"3cd2",

X"6e2a",

X"3535",

X"6f12",

X"372b",

X"9d5f",

X"8ce4",

X"88e1",

X"c5e5",

X"6381",

X"5891",

X"3e15",

X"7662",

X"0277",

X"8a96",

X"91f1",

X"8e50",

X"f7ef",

X"7065",

X"40d2",

X"48b3",

X"5e91",

X"e2e5",

X"996c",

X"9131",

X"8c2f",

X"1f8d",

X"7982",

X"3a61",

X"55ca",

X"4d57",

X"c61b",

X"959c",

X"8aa6",

X"a32f",

X"44f7",

X"6dde",

X"386a",

X"64e1",

X"173d",

X"a5ae",

X"9852",

X"8b57",

X"be73",

X"5a38",

X"6c81",

X"4001",

X"6417",

X"183e",

X"9d5e",

X"a322",

X"879b",

X"ed86",

X"7023",

X"5435",

X"49aa",

X"5bd5",

X"e695",

X"9edf",

X"a92c",

X"9261",

X"ffbd",

X"7097",

X"4880",

X"509b",

X"4177",

X"c3f6",

X"b0d3",

X"aa96",

X"a7d6",

X"27e8",

X"6102",

X"3d0d",

X"5238",

X"1c31",

X"b412",

X"b9f0",

X"a7a8",

X"c41a",

X"476c",

X"513e",

X"3c24",

X"4a03",

X"f449",

X"b812",

X"c4ea",

X"ac1a",

X"e451",

X"477e",

X"43a7",

X"3bc8",

X"42c4",

X"f2e7",

X"bafb",

X"c92e",

X"b73d",

X"00aa",

X"4778",

X"33e0",

X"403f",

X"19df",

X"c988",

X"c9a9",

X"c9f8",

X"c472",

X"0a87",

X"4103",

X"2cf4",

X"4ced",

X"0b39",

X"b72e",

X"ba40",

X"d8c2",

X"eb34",

X"3582",

X"27fb",

X"114e",

X"1bb7",

X"ff3c",

X"e525",

X"f1ec",

X"cb55",

X"d240",

X"107f",

X"20bf",

X"4053",

X"3e11",

X"fc6d",

X"c16f",

X"c9c3",

X"c7ca",

X"171b",

X"3e7c",

X"357d",

X"10df",

X"01d6",

X"cb95",

X"e84b",

X"fbec",

X"f65d",

X"f23b",

X"07d3",

X"fd32",

X"1fc6",

X"1fc2",

X"fa44",

X"f1e7",

X"cab9",

X"d166",

X"0dfb",

X"3790",

X"2e46",

X"2f24",

X"f004",

X"cc92",

X"dc15",

X"fef6",

X"05a8",

X"2dd1",

X"062c",

X"0341",

X"fba5",

X"f6d0",

X"036a",

X"06ce",

X"de5a",

X"e2d4",

X"fbb6",

X"118f",

X"2b31",

X"255f",

X"000e",

X"d668",

X"da3f",

X"e7f6",

X"1da7",

X"2842",

X"215c",

X"fd3a",

X"f265",

X"d92f",

X"0346",

X"0cc0",

X"0b26",

X"f9d2",

X"f9a8",

X"f4af",

X"12c7",

X"154e",

X"086d",

X"eff1",

X"e254",

X"e7d1",

X"0741",

X"1b11",

X"1994",

X"112f",

X"ebf9",

X"e6f5",

X"f529",

X"0ac7",

X"0c8f",

X"0ceb",

X"f23c",

X"ef0b",

X"f6bd",

X"049f",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

X"0000",

```
            X"0000"




);
constant ROM2 : rom_type :=
(


  X"ff9d",
   X"d2c4",
   X"f9ea",
   X"f73e",
   X"ee39",
   X"ed68",
   X"fc1c",
   X"f0c5",
   X"297e",
   X"fbe9",
   X"44e1",
   X"064f",
   X"2624",
   X"f409",
   X"ef34",
```

X"e789",

X"c9af",

X"01e7",

X"d6b2",

X"3b26",

X"f5c3",

X"4a4a",

X"00be",

X"2b18",

X"ebef",

X"e947",

X"e75d",

X"c400",

X"04c4",

X"cdc9",

X"2c0f",

X"04b1",

X"3b0e",

X"1e83",

X"1549",

X"15f9",

X"db04",

X"f2b7",

X"b6f6",

X"fefe",

X"d47c",

X"2d1b",

X"0265",

X"3877",

X"267d",

X"1811",

X"1353",

X"d582",

X"0230",

X"b7be",

X"f462",

X"db23",

X"139e",

X"15fc",

X"19d2",

X"3879",

X"05ea",

X"2a1d",

X"cf60",

X"0147",

X"b43d",

X"0048",

X"cfbf",

X"1a6a",

X"0d58",

X"23d4",

X"f20b",

X"de43",

X"da44",

X"c9d9",

X"f1d5",

X"d634",

X"2cb6",

X"fdae",

X"4da2",

X"0c27",

X"353d",

X"02d0",

X"ef36",

X"e0e9",

X"bfea",

X"f82f",

X"c5b2",

X"2810",

X"fdc0",

X"4188",

X"20fb",

X"2947",

X"2209",

X"e6e5",

X"f4f3",

X"b219",

X"f8b6",

X"c710",

X"2011",

X"f842",

X"2e3e",

X"264a",

X"2105",

X"2334",

X"e021",

X"0432",

X"b897",

X"fefa",

X"c48f",

X"129b",

X"0185",

X"2dea",

X"287a",

X"1685",

X"24ca",

X"e363",

X"f2eb",

X"c225",

X"ec20",

X"cbab",

X"f28e",

X"0998",

X"03e7",

X"4739",

X"0988",

X"4fc0",

X"ec8d",

X"1795",

X"c120",

X"e551",

X"c851",

X"ea39",

X"0814",

X"0ba4",

X"46a3",

X"0812",

X"4466",

X"d003",

X"034d",

X"b474",

X"e600",

X"c44f",

X"ec81",

X"1d51",

X"0e88",

X"511d",

X"1132",

X"445f",

X"e434",

X"070f",

X"c595",

X"d714",

X"d416",

X"e482",

X"1ac1",

X"ff71",

X"4eac",

X"0f17",

X"41e4",

X"ead6",

X"0636",

X"cd20",

X"cec3",

X"e1c6",

X"d484",

X"1d68",

X"f902",

X"49e2",

X"10c3",

X"3aa7",

X"0912",

X"f914",

X"ea1c",

X"ca2a",

X"f181",

X"c791",

X"1bd0",

X"f298",

X"4176",

X"129f",

X"3209",

X"efb2",

X"ea1e",

X"cd05",

X"d005",

X"e3c0",

X"dd25",

X"212a",

X"04f6",

X"4594",

X"1dac",

X"32f8",

X"fd9c",

X"eedb",

X"df8b",

X"ca65",

X"ea8d",

X"d4a4",

X"2308",

X"0490",

X"43df",

X"15a6",

X"2d84",

X"090d",

X"e4dc",

X"ddda",

X"c506",

X"f1f6",

X"cee4",

X"1895",

X"05b5",

X"3ad6",

X"21e6",

X"2364",

X"0aef",

X"eec4",

X"ea39",

X"c9c1",

X"f139",

X"d60d",

X"1d81",

X"037a",

X"355d",

X"1afc",

X"2014",

X"154f",

X"e4a1",

X"f287",

X"cdad",

X"f531",

X"d0d2",

X"072f",

X"083f",

X"1e2f",

X"2c65",

X"13ab",

X"28f8",

X"eb15",

X"0711",

X"ccd4",

X"f309",

X"dacc",

X"05fb",

X"00e9",

X"1ab9",

X"24ed",

X"139c",

X"1a3d",

X"e540",

X"fbcb",

X"d011",

X"f258",

X"daff",

X"01e5",

X"0d2d",

X"14c5",

X"2bb8",

X"0c9e",

X"2822",

X"eb45",

X"00af",

X"d74d",

X"eede",

X"e373",

X"f980",

X"0e25",

X"0d68",

X"2a74",

X"0e1f",

X"20b1",

X"eda7",

X"fff7",

X"d9aa",

X"edb0",

X"e2f1",

X"f053",

X"00f0",

X"f4f3",

X"1312",

X"f901",

X"2107",

X"101d",

X"1e60",

X"0724",

X"0d7b",

X"041f",

X"dbb8",

X"ef09",

X"ded3",

X"0264",

X"e644",

X"139b",

X"1488",

X"1fb3",

X"097a",

X"14dd",

X"094c",

X"dbb4",

X"eb55",

X"e25b",

X"fdc3",

X"e17a",

X"07be",

X"16bb",

X"21f8",

X"11d5",

X"1018",

X"0e2b",

X"ec8c",

X"f006",

X"da61",

X"fca1",

X"dfed",

X"06bc",

X"14ec",

X"1b44",

X"1566",

X"1313",

X"15d0",

X"ed04",

X"f038",

X"dd51",

X"f89e",

X"e11e",

X"f7ec",

X"080e",

X"1df8",

X"1a11",

X"1297",

X"1e61",

X"f809",

X"edc9",

X"d828",

X"f719",

X"dffc",

X"f372",

X"115e",

X"1f07",

X"1ac3",

X"1266",

X"1cb4",

X"f59c",

X"e97e",

X"d0a4",

X"e76b",

X"dc62",

X"e744",

X"1f15",

X"2df9",

X"34a0",

X"1d14",

X"3562",

X"094a",

X"e699",

X"c57a",

X"dc06",

X"d59f",

X"dfd5",

X"0c61",

X"28b1",

X"3251",

X"1f40",

X"34d8",

X"04b6",

X"e07f",

X"c925",

X"d872",

X"da07",

X"d60d",

X"0ca1",

X"2cdc",

X"37ad",

X"223b",

X"3394",

X"157d",

X"f030",

X"cbd1",

X"d1e9",

X"d8c8",

X"d422",

X"0855",

X"2891",

X"3940",

X"2272",

X"3587",

X"1c9a",

X"e9c5",

X"c69a",

X"c98c",

X"d00e",

X"c217",

X"fb05",

X"3074",

X"47ac",

X"316f",

X"40ab",

X"308d",

X"f2b3",

X"c349",

X"be6b",

X"cd8b",

X"bc47",

X"ef89",

X"2a88",

X"493e",

X"3567",

X"42e0",

X"2bc4",

X"eed2",

X"be6b",

X"bcc3",

X"c810",

X"ba33",

X"ea9c",

X"2b95",

X"4b13",

X"3c8d",

X"431f",

X"3ffa",

X"0530",

X"c829",

X"b8c1",

X"c801",

X"bb45",

X"dd9d",

X"1487",

X"47f3",

X"3f00",

X"4164",

X"3f67",

X"0194",

X"c5d7",

X"b527",

X"ccad",

X"b9bc",

X"dc13",

X"2105",

X"4b64",

X"3c5e",

X"3e34",

X"4600",

X"0f72",

X"cc19",

X"b619",

X"cd4d",

X"bdf6",

X"cff2",

X"0b75",

X"43d7",

X"3de2",

X"38f2",

X"44d3",

X"1abf",

X"d429",

X"b772",

X"c8da",

X"bebf",

X"ce2b",

X"0aa7",

X"40ea",

X"4714",

X"38fe",

X"4990",

X"1a82",

X"d646",

X"b3c8",

X"c6f4",

X"c06f",

X"c695",

X"fc26",

X"3b25",

X"447c",

X"3995",

X"4457",

X"23e7",

X"e2ac",

X"b769",

X"c54d",

X"c2b8",

X"c5b4",

X"fe3d",

X"3951",

X"478e",

X"3635",

X"4525",

X"2f4d",

X"ed9c",

X"b7e4",

X"c5be",

X"c741",

X"c345",

X"ef1c",

X"2eec",

X"4605",

X"3632",

X"3fb2",

X"2cb6",

X"ed03",

X"b9ef",

X"c50f",

X"c689",

X"bfd7",

X"ecf9",

X"317c",

X"4806",

X"36fa",

X"414e",

X"360f",

X"f9f3",

X"c13e",

X"c39e",

X"c9c6",

X"c1c2",

X"e482",

X"1f44",

X"4289",

X"3665",

X"3af8",

X"3137",

X"fa2e",

X"c5f0",

X"c3a5",

X"caf1",

X"c29c",

X"e51b",

X"2402",

X"4494",

X"3643",

X"3863",

X"3473",

X"0595",

X"caec",

X"c59a",

X"d060",

X"c6eb",

X"e02c",

X"1322",

X"3b2e",

X"333b",

X"343f",

X"338d",

X"0d5c",

X"d525",

X"c7d0",

X"d4ac",

X"ca93",

X"df95",

X"14f7",

X"3a17",

X"2f75",

X"2fb8",

X"3273",

X"0c2c",

X"d524",

X"cb69",

X"d8ef",

X"d038",

X"ddf4",

X"0b91",

X"3179",

X"2f0c",

X"2a29",

X"2e79",

X"10c1",

X"e0f6",

X"cf7d",

X"da3e",

X"d28c",

X"e0f7",

X"0b4e",

X"2d7e",

X"2be9",

X"25f3",

X"2e61",

X"1306",

X"dedc",

X"d2d4",

X"dcae",

X"d8e4",

X"dee6",

X"00a7",

X"28d6",

X"289b",

X"24aa",

X"28d8",

X"14ea",

X"eb9c",

X"d5c5",

X"de56",

X"d995",

X"de85",

X"0315",

X"2655",

X"2824",

X"214f",

X"2921",

X"180a",

X"f19b",

X"d7ac",

X"e0ba",

X"df04",

X"df9f",

X"fa49",

X"1e24",

X"256e",

X"2120",

X"240c",

X"0ce3",

X"deb1",

X"c526",

X"c544",

X"c1e1",

X"e0ed",

X"2320",

X"44e1",

X"45e5",

X"546f",

X"3821",

X"fe9c",

X"bf66",

X"bb7b",

X"b07d",

X"b8fb",

X"decc",

X"213f",

X"4786",

X"49a5",

X"5775",

X"2f71",

X"f7f8",

X"c1ff",

X"bec6",

X"b1a0",

X"affa",

X"dc9c",

X"2234",

X"4c87",

X"434a",

X"5233",

X"3b78",

X"08e6",

X"c5cc",

X"bd3f",

X"b838",

X"b3bf",

X"dd58",

X"1bfd",

X"4874",

X"3f01",

X"5042",

X"3d59",

X"0d66",

X"ccb1",

X"bb7e",

X"bfba",

X"b00f",

X"d374",

X"06de",

X"4a80",

X"3f31",

X"4df2",

X"3c46",

X"18f6",

X"ddc4",

X"b38c",

X"c5d6",

X"b11e",

X"d4e7",

X"04b0",

X"437f",

X"3c1e",

X"4a01",

X"3caa",

X"1434",

X"d57e",

X"b3a0",

X"bdd7",

X"a663",

X"db80",

X"096c",

X"5ce8",

X"4290",

X"555c",

X"5281",

X"17f3",

X"d69a",

X"9e1f",

X"c11a",

X"99ff",

X"c8c8",

X"00c7",

X"512e",

X"4aae",

X"52b1",

X"54ba",

X"1540",

X"d1df",

X"a45d",

X"c1fc",

X"9cd8",

X"c049",

X"0496",

X"4f3f",

X"50f0",

X"463b",

X"6532",

X"1d95",

X"eb4e",

X"9ff8",

X"c106",

X"a690",

X"bb67",

X"f0b4",

X"3738",

X"5b57",

X"3f23",

X"67bb",

X"170d",

X"f0d7",

X"9825",

X"b80c",

X"9836",

X"ac20",

X"f74d",

X"40de",

X"70af",

X"4590",

X"7a38",

X"2aab",

X"f1cc",

X"96b2",

X"aaa6",

X"9e26",

X"9e9a",

X"f5c2",

X"36a4",

X"7a81",

X"43ea",

X"8000",

X"2ce8",

X"ee69",

X"92f7",

X"a590",

X"9e77",

X"97f4",

X"ee27",

X"2ac6",

X"7f49",

X"49fe",

X"7c70",

X"4b67",

X"018c",

X"ae5c",

X"9123",

X"b066",

X"8534",

X"fbfc",

X"1744",

X"7e8d",

X"4a84",

X"7899",

X"4a60",

X"f76e",

X"b52b",

X"9118",

X"ba57",

X"8221",

X"ebde",

X"1747",

X"7fde",

X"449e",

X"6f2d",

X"55c4",

X"0428",

X"c493",

X"8828",

X"c7cb",

X"82f3",

X"dd09",

X"04ef",

X"6785",

X"528c",

X"5cad",

X"5f89",

X"fdc2",

X"ded8",

X"89ee",

X"c77f",

X"7fef",

X"e86f",

X"0d31",

X"6056",

X"56d4",

X"53f1",

X"6e0d",

X"fc5f",

X"db3d",

X"86b4",

X"c913",

X"8a45",

X"c995",

X"09aa",

X"51ee",

X"63a2",

X"49d6",

X"6db7",

X"01f7",

X"ece8",

X"8907",

X"c48d",

X"9105",

X"cf02",

X"0f8b",

X"4ae4",

X"6930",

X"3f53",

X"7885",

X"0caf",

X"f34e",

X"9587",

X"bcdc",

X"a45b",

X"acf7",

X"0d57",

X"2cc0",

X"6ef2",

X"3464",

X"78f6",

X"0edc",

X"f0f6",

X"a3d7",

X"b61e",

X"a8ae",

X"af16",

X"095a",

X"2fff",

X"71f7",

X"362b",

X"7812",

X"11f5",

X"ffe7",

X"a10b",

X"b032",

X"b551",

X"a418",

X"0b08",

X"1ad9",

X"6bc5",

X"3838",

X"692b",

X"2493",

X"ed74",

X"b90a",

X"ab6c",

X"bcd5",

X"9e20",

X"0764",

X"1fe5",

X"6a02",

X"3310",

X"66d0",

X"3520",

X"fb83",

X"bda4",

X"a6c5",

X"cc1b",

X"9c87",

X"0042",

X"0e65",

X"628f",

X"358c",

X"58fa",

X"328b",

X"f729",

X"d3f0",

X"a804",

X"cf84",

X"a115",

X"04b5",

X"0d91",

X"5339",

X"3513",

X"4e15",

X"3d33",

X"f791",

X"d88e",

X"ae19",

X"d9b1",

X"a952",

X"f4f4",

X"08e4",

X"4950",

X"35b2",

X"3f0b",

X"368f",

X"fa5a",

X"e599",

X"ae7a",

X"de4d",

X"ad40",

X"f4db",

X"08fc",

X"4269",

X"3231",

X"3ba1",

X"3be4",

X"f5d4",

```
    X"e389",
    X"b77a",
    X"e29a",
    X"b52d",
    X"eb1d",
    X"0aba",
    X"2e40",
    X"37e2",
    X"2c87",
    X"4046",
    X"fb26",
    X"ef56",
    X"bc1b",
    X"df9b",
    X"ba9b",
    X"e954",
    X"0992",
    X"2b1a"

);

  signal test_mode: std_logic :='1';              -- Output a sine wave
  signal audio_request: std_logic:='1';
  signal data: unsigned(15 downto 0):= "0000000000000000";
```

```
  signal audio_clock : unsigned(1 downto 0) := "00";
  signal data_w : unsigned (15 downto 0);
  signal output_flag : std_logic:='0';
  signal output_flag2 : std_logic := '0';
  signal num: integer;
---------------
begin
process (clk)
 begin
  if rising_edge(clk) then
    audio_clock <= audio_clock + "1";
  end if;
 end process;

  -- LRCK divider
  -- Audio chip main clock is 18.432MHz / Sample rate 48KHz
  -- Divider is 18.432 MHz / 48KHz = 192 (X"C0")   10b
  -- Left justify mode set by I2C controller

 process (audio_clock(1))
 begin
  if rising_edge(audio_clock(1)) then
    if reset_n = '0' then
      lrck_divider <= (others => '0');
```

```
   elsif lrck_divider = "100001000000" THEN--X"BF"  then       -- "C0" minus 1
       lrck_divider <= "000000000000";
   else
     lrck_divider <= lrck_divider + 1;
   end if;
 end if;
end process;

process (clk)       --changed for clock frequency
begin
 if rising_edge(audio_clock(1)) then
   if reset_n = '0' then
     bclk_divider <= (others => '0');
   elsif bclk_divider = X"B" or set_lrck = '1'  then
     bclk_divider <= X"0";
   else
     bclk_divider <= bclk_divider + 1;
   end if;
 end if;
end process;

set_lrck <= '1' when lrck_divider = X"BF" else '0';

process (audio_clock(1))
```

```
begin
  if rising_edge(audio_clock(1)) then
    if reset_n = '0' then
      lrck <= '0';
    elsif set_lrck = '1' then
      lrck <= not lrck;
    end if;
  end if;
end process;

-- BCLK divider
set_bclk <= '1' when bclk_divider(3 downto 0) = "0101" else '0';
clr_bclk <= '1' when bclk_divider(3 downto 0) = "1011" else '0';

process (audio_clock(1))
begin
  if rising_edge(audio_clock(1)) then
    if reset_n = '0' then
      bclk <= '0';
    elsif set_lrck = '1' or clr_bclk = '1' then
      bclk <= '0';
    elsif set_bclk = '1' then
      bclk <= '1';
    end if;
```

```
  end if;
end process;

-- Audio data shift output
process (audio_clock(1))
begin
  if rising_edge(audio_clock(1)) then
    if reset_n = '0' then
      shift_out <= (others => '0');
    elsif set_lrck = '1' then
      if test_mode = '1' then
        shift_out <= sin_out;
      else
        shift_out <= data;
      end if;
    elsif clr_bclk = '1' then
      shift_out <= shift_out (14 downto 0) & '0';
    end if;
  end if;
end process;

  -- Audio outputs

  AUD_ADCLRCK  <= lrck;
```

```
   AUD_DACLRCK  <= lrck;
   AUD_DACDAT   <= shift_out(15);
   AUD_BCLK     <= bclk;

   -- Self test with Sin wave
--process(audio_clock(1))
--begin
--    if rising_edge(audio_clock(1)) then
--      if chipselect = '1' then
--         if address = "11111"  then                        -- vertical 4
--            if write = '1' then
--             data_w<= writedata(15 downto 0) ;
--             num<=to_integer(data_w);
--             output_flag<='1';
--            end if;
--         end if;
--      end if;
--      if sin_counter =x"544" then
--        output_flag<1='0';
--      end if;
--    end if;
--end process;

   process(clk)
```

```
    begin
      if rising_edge(clk) then
-------------------------------------------
    if chipselect = '1' then
         if address = "11111"  then                          -- vertical 4
            if write = '1' then
             data_w <= writedata(15 downto 0) ;
             num<=to_integer(data_w);
             output_flag <= '1';

            end if;
          end if;
      end if;
    if output_flag2 = '1' then
       output_flag <= '0';
    end if;
     end if;
  end process;
---------------------------------------
process(audio_clock(1))
   begin
    if rising_edge(audio_clock(1)) then
      if reset_n = '0' then
           sin_counter <= (others => '0');
```

```
        elsif output_flag = '1' then
         if lrck_lat = '1' and lrck = '0'  then
--          if sin_counter = X"544" then
--            sin_counter <= X"000";
--          else
            sin_counter <= sin_counter + 1;
--          end if;
        end if;
      end if;
--------------------------------
      if sin_counter = x"398" then
        output_flag2 <= '1'; sin_counter <= x"000";
      elsif sin_counter /= x"398" then
        output_flag2 <= '0';
      end if;
---------------------------------
    end if;
    end process;


    process(audio_clock(1))
    begin
      if rising_edge(audio_clock(1)) then
        lrck_lat <= lrck;
      end if;
```

```
    end process;


    process (audio_clock(1))
    begin
     if rising_edge(audio_clock(1)) then
       if lrck_lat = '1' and lrck = '0' then
         audio_request <= '1';
       else
         audio_request <= '0';
       end if;
     end if;
    end process;
    with num select sin_out<=
            ROM1(TO_INTEGER(SIN_COUNTER)) when 1,
            ROM2(TO_INTEGER(SIN_COUNTER)) when 2,
            null when others;
--  with sin_counter select sin_out <=
--    X"0000" when "000000",
--    X"10b4" when "000001",
--    X"2120" when "000010",
--    X"30fb" when "000011",
--    X"3fff" when "000100",
--    X"4deb" when "000101",
--    X"5a81" when "000110",
```

```
--    X"658b" when "000111",
--    X"6ed9" when "001000",
--    X"7640" when "001001",
--    X"7ba2" when "001010",
--    X"7ee6" when "001011",
--    X"7fff" when "001100",
--    X"7ee6" when "001101",
--    X"7ba2" when "001110",
--    X"7640" when "001111",
--    X"6ed9" when "010000",
--    X"658b" when "010001",
--    X"5a81" when "010010",
--    X"4deb" when "010011",
--    X"3fff" when "010100",
--    X"30fb" when "010101",
--    X"2120" when "010110",
--    X"10b4" when "010111",
--    X"0000" when "011000",
--    X"ef4b" when "011001",
--    X"dee0" when "011010",
--    X"cf05" when "011011",
--    X"c001" when "011100",
--    X"b215" when "011101",
--    X"a57e" when "011110",
```

```
--    X"9a74" when "011111",
--    X"9127" when "100000",
--    X"89bf" when "100001",
--    X"845d" when "100010",
--    X"8119" when "100011",
--    X"8000" when "100100",
--    X"8119" when "100101",
--    X"845d" when "100110",
--    X"89bf" when "100111",
--    X"9127" when "101000",
--    X"9a74" when "101001",
--    X"a57e" when "101010",
--    X"b215" when "101011",
--    X"c000" when "101100",
--    X"cf05" when "101101",
--    X"dee0" when "101110",
--    X"ef4b" when "101111",
--    X"0000" when others;

end architecture;

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
entity ice_cream_red_down is

  port(

clk   : in std_logic;

addr  : in integer;

data  : out std_logic_vector (4 downto 0)

  );

end ice_cream_red_down;



architecture rtl of ice_cream_red_down is



type rom_type is array (0 to 1023) of std_logic_vector(4 downto 0);   -- denote tile
content



constant ROM :rom_type :=  (

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00000","00000","00000","00111","10000","10001","10001","01011","00001","
00000","00000","00000","00000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00000","00000","00000","01011","10111","11101","11101","11010","01110","
00011","00000","00000","00000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00000","00000","00000","00010","01110","10110","11010","11110","10011","
00111","00000","00000","00000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00000","00000","00001","01010","10010","10011","11000","11111","11101","
10011","00110","00000","00000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",
```

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00000","00001","01100","11001","11110","11101","11110","11111","11111","
10101","01001","00000","00000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00001","01010","10111","11101","11111","11111","11111","11111","11111","
11010","10010","01001","00011","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00101","10011","11101","11111","11111","11111","11111","11111","11111","
11111","11001","10011","01000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00101","10011","11101","11111","11111","11111","11111","11111","11110","
11111","11000","10011","01000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","001
00","01110","10011","11100","11110","11110","11110","11110","11110","11110","
11110","11000","10011","10000","00110","00000","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00001","00100","011
10","10110","10110","11010","11100","11110","11110","11110","11110","11110","
11011","11000","10110","10110","10001","00100","00000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00110","10010","110
00","11111","11111","11000","11010","11110","11110","11110","11110","11101","
10111","11011","11111","11111","11010","01110","00001","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00010","01110","10110","110
10","11111","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11100","10100","01000","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","01011","10111","11111","111
10","11110","11111","11111","11111","11111","11111","11111","11111","11111","
11111","11111","11111","11111","11111","11101","10010","00101","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","01100","10111","11110","111
10","11110","11110","11111","11111","11111","11111","11111","11111","11111","

11111","11111","11111","11110","11110","11101","10011","00101","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","01011","10110","11100","111
01","11110","11110","11111","11111","11111","11111","11111","11111","11111","
11111","11110","11110","11110","11110","11100","10010","00101","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00010","01110","10101","10110","110
01","11101","11110","11110","11110","11110","11110","11110","11110","11110","
11110","11110","11110","11110","11010","10110","10100","01000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00011","01101","11000","11100","11011","111
01","11111","11101","11101","11110","11110","11110","11110","11110","11110","
11101","11101","11101","11101","11100","11100","11011","10011","01001","0001
0","00000","00000","00000",

"00000","00000","00000","00011","01010","01100","01100","01100","01100","100
10","11101","11100","11100","11100","11101","11111","11111","11110","11100","
11100","11100","11100","10001","01100","01100","01100","01011","10000","1001
0","00110","00000","00000",

"00000","00000","00000","00100","01010","01011","01011","01011","01011","010
11","01100","01100","01100","01011","10010","11111","11111","11100","01100","
01100","01100","01100","01011","01011","01011","01011","01010","10000","1010
0","01000","00000","00000",

"00000","00001","01010","01100","01001","01001","01001","01001","01011","010
11","01011","01011","01011","01010","10001","11111","11111","11011","01011","
01011","01011","01011","01011","01011","01010","01001","01000","10000","1010
0","01111","00101","00000",

"00000","00100","10011","10010","10000","10110","11000","10011","01000","010
01","01001","01001","01001","01000","01111","11111","11111","11011","01001","
01001","01001","01001","01001","01000","01011","11000","11000","10101","1010
0","10011","01010","00000",

"00000","00100","10010","10010","10100","11100","11111","11101","11000","110
01","10110","01001","10101","11001","11010","11111","11111","11110","11001","
11001","01111","00111","10011","11001","11010","11111","11111","11000","1010
0","10011","01010","00000",

"00000","00100","10010","10010","10010","10101","11011","11111","11111","111
11","11011","01001","11011","11111","11111","11111","11111","11111","11111","
11111","10010","00111","11000","11111","11111","11110","10111","10100","1001
0","10011","01010","00000",

"00000","00100","10011","10010","10010","10100","11001","11110","11110","111
11","11011","01000","11000","11100","11100","11100","11100","11100","11100","
11100","10000","00110","10111","11111","11110","11101","10110","10011","1000
1","10010","01010","00000",

"00000","00100","10011","10010","10010","01101","10100","11011","11110","111
10","11100","10000","11010","11101","11101","11101","11101","11101","11101","
11101","10101","01111","11010","11111","11101","11001","01110","10000","1001
1","10011","01010","00000",

"00000","00001","01001","01011","01011","01100","01100","10001","11001","110
11","11101","11001","11110","11101","11010","11010","11010","11010","11010","
11111","11100","11001","11100","11100","10111","01101","01100","01100","0101
1","01100","00101","00000",

"00000","00000","00000","00011","00010","01110","01010","00111","01101","100
01","11000","10101","11110","11100","11001","11001","11001","11001","11010","
11111","11001","10101","11000","10100","01011","00101","01111","01000","0000
1","00011","00000","00000",

"00000","00000","00000","00001","00100","00100","00011","00100","00011","010
01","10011","10011","11001","11010","11010","11010","11010","11010","11010","
11010","10110","10011","10011","01100","00011","00100","00011","00101","0001
1","00001","00000","00000",

"00000","00000","00000","00000","00001","00011","00101","00101","00101","010
10","10010","10010","10100","10101","10100","10100","10100","10100","10100","
10101","10011","10001","10010","01100","00101","00101","00100","00011","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00001","01001","01110","01110","100
01","10010","10010","10010","10010","10010","10010","10010","10010","10010","
10010","10010","10010","10010","10001","01111","01101","00101","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00001","00001","001
11","01101","01111","10010","10010","10010","10010","10010","10010","10010","
10010","10001","01110","01110","01001","00010","00001","00000","00000","0000
0","00000","00000","00000",

"00000","00000","00000","00000","00000","00000","00000","00000","00000","000
00","00000","00010","01010","01101","01101","01101","01101","01101","01101","
01101","00110","00001","00000","00000","00000","00000","00000","00000","0000
0","00000","00000","00000"

);

```
begin


  process(clk)
begin
  if rising_edge(clk) then
  data <= ROM(addr);
  end if;
  end process;


end rtl;
```

ROM for many ..... 72 files

ice_cream_controller

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity ice_cream_controller is
  port(
clk   : in std_logic;
address: in unsigned (4 downto 0);
vertical: in integer;
```

```
horizontal: in integer;

pixel_R: out std_logic_vector (4 downto 0);
pixel_G: out std_logic_vector (4 downto 0);
pixel_B: out std_logic_vector (4 downto 0)
  );

end ice_cream_controller;

architecture rtl of ice_cream_controller is
------------------------------------------------------------sprite_down----------------------------
component ice_cream_red_down
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component ice_cream_green_down
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
```

```
  );
end component;

component ice_cream_blue_down
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
----------------------------------------------------------sprite_dlf--------------------------------
component ice_cream_red_dlf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component ice_cream_green_dlf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
```

```
  );
end component;

component ice_cream_blue_dlf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
------------------------------------------------------------sprite_drf-------------------------
component ice_cream_red_drf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component ice_cream_green_drf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
```

```
  );
end component;

component ice_cream_blue_drf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

------------------------------------------------------------sprite_up----------------------------
component ice_cream_red_up
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component ice_cream_green_up
  port(
clk   : in std_logic;
addr  : in integer;
```

```
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_up
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_ulf--------------------------------
component ice_cream_red_ulf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_ulf
 port(
clk   : in std_logic;
addr  : in integer;
```

```
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_ulf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_urf-------------------------
component ice_cream_red_urf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_urf
 port(
clk   : in std_logic;
addr  : in integer;
```

```
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_urf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

------------------------------------------------------------sprite_left----------------------------
component ice_cream_red_left
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_left
 port(
clk   : in std_logic;
```

```
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_left
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_llf-------------------------------
component ice_cream_red_llf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_llf
 port(
clk   : in std_logic;
```

```
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_llf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_lrf-------------------------
component ice_cream_red_lrf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_lrf
 port(
clk   : in std_logic;
```

```
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_lrf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

------------------------------------------------------------sprite_right-----------------------------
component ice_cream_red_right
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_right
 port(
```

```
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_right
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_rlf--------------------------------
component ice_cream_red_rlf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_rlf
 port(
```

```
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_rlf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_rrf-------------------------
component ice_cream_red_rrf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_rrf
 port(
```

```
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_blue_rrf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

----------dead----------
component ice_cream_red_dead
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component ice_cream_green_dead
```

```
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;



component ice_cream_blue_dead
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

signal ver : integer;
signal hor : integer;
signal sprite_R_down,sprite_G_down,sprite_B_down : std_logic_vector(4 downto 0);
signal sprite_R_dlf,sprite_G_dlf,sprite_B_dlf : std_logic_vector(4 downto 0);
signal sprite_R_drf,sprite_G_drf,sprite_B_drf : std_logic_vector(4 downto 0);

signal sprite_R_up,sprite_G_up,sprite_B_up : std_logic_vector(4 downto 0);
signal sprite_R_ulf,sprite_G_ulf,sprite_B_ulf : std_logic_vector(4 downto 0);
```

```
signal sprite_R_urf,sprite_G_urf,sprite_B_urf : std_logic_vector(4 downto 0);



signal sprite_R_left,sprite_G_left,sprite_B_left : std_logic_vector(4 downto 0);

signal sprite_R_llf,sprite_G_llf,sprite_B_llf : std_logic_vector(4 downto 0);

signal sprite_R_lrf,sprite_G_lrf,sprite_B_lrf : std_logic_vector(4 downto 0);


signal sprite_R_right,sprite_G_right,sprite_B_right : std_logic_vector(4 downto 0);

signal sprite_R_rlf,sprite_G_rlf,sprite_B_rlf : std_logic_vector(4 downto 0);

signal sprite_R_rrf,sprite_G_rrf,sprite_B_rrf : std_logic_vector(4 downto 0);



signal sprite_R_dead, sprite_G_dead, sprite_B_dead : std_logic_vector(4 downto 0);


begin

----------------------------------------down ----------------------------------

ice_cream_R_down: ice_cream_red_down port map
(clk,ver*32+hor,sprite_R_down);

ice_cream_G_down: ice_cream_green_down port map
(clk,ver*32+hor,sprite_G_down);

ice_cream_B_down: ice_cream_blue_down port map
(clk,ver*32+hor,sprite_B_down);


ice_cream_R_dlf: ice_cream_red_dlf port map (clk,ver*32+hor,sprite_R_dlf);

ice_cream_G_dlf: ice_cream_green_dlf port map (clk,ver*32+hor,sprite_G_dlf);

ice_cream_B_dlf: ice_cream_blue_dlf port map (clk,ver*32+hor,sprite_B_dlf);
```

```
ice_cream_R_drf: ice_cream_red_drf port map (clk,ver*32+hor,sprite_R_drf);
ice_cream_G_drf: ice_cream_green_drf port map (clk,ver*32+hor,sprite_G_drf);
ice_cream_B_drf: ice_cream_blue_drf port map (clk,ver*32+hor,sprite_B_drf);

-----------------------------------------up--------------------------------
ice_cream_R_up: ice_cream_red_up port map (clk,ver*32+hor,sprite_R_up);
ice_cream_G_up: ice_cream_green_up port map (clk,ver*32+hor,sprite_G_up);
ice_cream_B_up: ice_cream_blue_up port map (clk,ver*32+hor,sprite_B_up);

ice_cream_R_ulf: ice_cream_red_ulf port map (clk,ver*32+hor,sprite_R_ulf);
ice_cream_G_ulf: ice_cream_green_ulf port map (clk,ver*32+hor,sprite_G_ulf);
ice_cream_B_ulf: ice_cream_blue_ulf port map (clk,ver*32+hor,sprite_B_ulf);

ice_cream_R_urf: ice_cream_red_urf port map (clk,ver*32+hor,sprite_R_urf);
ice_cream_G_urf: ice_cream_green_urf port map (clk,ver*32+hor,sprite_G_urf);
ice_cream_B_urf: ice_cream_blue_urf port map (clk,ver*32+hor,sprite_B_urf);

-----------------------------------------left--------------------------------
ice_cream_R_left: ice_cream_red_left port map (clk,ver*32+hor,sprite_R_left);
ice_cream_G_left: ice_cream_green_left port map (clk,ver*32+hor,sprite_G_left);
ice_cream_B_left: ice_cream_blue_left port map (clk,ver*32+hor,sprite_B_left);

ice_cream_R_llf: ice_cream_red_llf port map (clk,ver*32+hor,sprite_R_llf);
```

```
ice_cream_G_llf: ice_cream_green_llf port map (clk,ver*32+hor,sprite_G_llf);

ice_cream_B_llf: ice_cream_blue_llf port map (clk,ver*32+hor,sprite_B_llf);


ice_cream_R_lrf: ice_cream_red_lrf port map (clk,ver*32+hor,sprite_R_lrf);

ice_cream_G_lrf: ice_cream_green_lrf port map (clk,ver*32+hor,sprite_G_lrf);

ice_cream_B_lrf: ice_cream_blue_lrf port map (clk,ver*32+hor,sprite_B_lrf);


------------------------------------------------right----------------------------

ice_cream_R_right: ice_cream_red_right port map (clk,ver*32+hor,sprite_R_right);

ice_cream_G_right: ice_cream_green_right port map
(clk,ver*32+hor,sprite_G_right);

ice_cream_B_right: ice_cream_blue_right port map (clk,ver*32+hor,sprite_B_right);


ice_cream_R_rlf: ice_cream_red_rlf port map (clk,ver*32+hor,sprite_R_rlf);

ice_cream_G_rlf: ice_cream_green_rlf port map (clk,ver*32+hor,sprite_G_rlf);

ice_cream_B_rlf: ice_cream_blue_rlf port map (clk,ver*32+hor,sprite_B_rlf);


ice_cream_R_rrf: ice_cream_red_rrf port map (clk,ver*32+hor,sprite_R_rrf);

ice_cream_G_rrf: ice_cream_green_rrf port map (clk,ver*32+hor,sprite_G_rrf);

ice_cream_B_rrf: ice_cream_blue_rrf port map (clk,ver*32+hor,sprite_B_rrf);



------------------------------------------------dead----------------------------


ice_cream_R_dead: ice_cream_red_dead port map (clk,ver*32+hor,sprite_R_dead);
```

```
ice_cream_G_dead: ice_cream_green_dead port map
(clk,ver*32+hor,sprite_G_dead);

ice_cream_B_dead: ice_cream_blue_dead port map (clk,ver*32+hor,sprite_B_dead);









process(clk)

begin

if rising_edge(clk) then

  hor <= horizontal;

  ver <= vertical;

end if;

end process;


process(clk)

begin

if rising_edge(clk) then

case address is

          ----------------------------------------down----------------------------------------

 when "00000" =>                                                                    -- down

--  if sprite_R_down /= "00000" then

   pixel_R <= sprite_R_down;

--  else

--  pixel_R <= "11111";
```

```
--   end if;


--   if sprite_G_down /= "00000" then
     pixel_G <= sprite_G_down;
--   else
--   pixel_G <= "11111";
--   end if;


--   if sprite_B_down /= "00000" then
     pixel_B <= sprite_B_down;
--   else
--   pixel_B <= "11111";
--   end if;


   when "00001" =>                                          -- dlf
--   if sprite_R_dlf /= "00000" then
     pixel_R <= sprite_R_dlf;
--   else
--   pixel_R <= "11111";
--   end if;


--   if sprite_G_dlf /= "00000" then
     pixel_G <= sprite_G_dlf;
--   else
```

```
--    pixel_G <= "11111";
--    end if;

--    if sprite_B_dlf /= "00000" then
      pixel_B <= sprite_B_dlf;
--    else
--    pixel_B <= "11111";
 --   end if;

 when "00010" =>                          -- drf
--     if sprite_R_drf /= "00000" then
      pixel_R <= sprite_R_drf;
 --   else
 --   pixel_R <= "11111";
 --   end if;

 --   if sprite_G_drf /= "00000" then
      pixel_G <= sprite_G_drf;
--    else
--    pixel_G <= "11111";
 --   end if;

 --   if sprite_B_drf /= "00000" then
      pixel_B <= sprite_B_drf;
```

```
--   else
--   pixel_B <= "11111";
--   end if;

------------------------------------------------------------------up--------------------------
 when "00011" =>
--   if sprite_R_up /= "00000" then
    pixel_R <= sprite_R_up;
--   else
--   pixel_R <= "11111";
--   end if;

--   if sprite_G_up /= "00000" then
    pixel_G <= sprite_G_up;
--   else
--   pixel_G <= "11111";
--   end if;

--   if sprite_B_up /= "00000" then
    pixel_B <= sprite_B_up;
--   else
--   pixel_B <= "11111";
--   end if;
```

```
 when "00100" =>
--   if sprite_R_ulf /= "00000" then
   pixel_R <= sprite_R_ulf;
--   else
--   pixel_R <= "11111";
--   end if;

--   if sprite_G_ulf /= "00000" then
   pixel_G <= sprite_G_ulf;
--   else
--   pixel_G <= "11111";
--   end if;

--   if sprite_B_ulf /= "00000" then
   pixel_B <= sprite_B_ulf;
--   else
--   pixel_B <= "11111";
--   end if;

 when "00101" =>                        -- urf
--    if sprite_R_urf /= "00000" then
   pixel_R <= sprite_R_urf;
--   else
--   pixel_R <= "11111";
```

```
--    end if;


 --   if sprite_G_urf /= "00000" then
     pixel_G <= sprite_G_urf;
--    else
--    pixel_G <= "11111";
 --   end if;


  --  if sprite_B_urf /= "00000" then
     pixel_B <= sprite_B_urf;
--    else
--    pixel_B <= "11111";
 --   end if;


------------------------------------------------------------------left---------------------------
 when "00110" =>
 --   if sprite_R_left /= "00000" then
     pixel_R <= sprite_R_left;
 --   else
 --   pixel_R <= "11111";
--    end if;


 --   if sprite_G_left /= "00000" then
     pixel_G <= sprite_G_left;
```

```
-- else
-- pixel_G <= "11111";
-- end if;

-- if sprite_B_left /= "00000" then
  pixel_B <= sprite_B_left;
--  else
--  pixel_B <= "11111";
--  end if;

 when "00111" =>
 -- if sprite_R_llf /= "00000" then
  pixel_R <= sprite_R_llf;
--  else
--  pixel_R <= "11111";
--  end if;

--  if sprite_G_llf /= "00000" then
  pixel_G <= sprite_G_llf;
--  else
--  pixel_G <= "11111";
--  end if;

 -- if sprite_B_llf /= "00000" then
```

```
   pixel_B <= sprite_B_llf;
--   else
--   pixel_B <= "11111";
 --  end if;

when "01000" =>                          -- lrf
--    if sprite_R_lrf /= "00000" then
   pixel_R <= sprite_R_lrf;
--   else
--   pixel_R <= "11111";
--   end if;

--   if sprite_G_lrf /= "00000" then
   pixel_G <= sprite_G_lrf;
--   else
--   pixel_G <= "11111";
--   end if;

 --  if sprite_B_lrf /= "00000" then
   pixel_B <= sprite_B_lrf;
--   else
--  pixel_B <= "11111";
--   end if;
```

```
-----------------------------------------------------------------right----------------------------
 when "01001" =>
--   if sprite_R_right /= "00000" then
    pixel_R <= sprite_R_right;
--   else
--   pixel_R <= "11111";
--   end if;

 --  if sprite_G_right /= "00000" then
    pixel_G <= sprite_G_right;
--   else
--   pixel_G <= "11111";
 --  end if;

 --  if sprite_B_right /= "00000" then
    pixel_B <= sprite_B_right;
--   else
--   pixel_B <= "11111";
 --  end if;

 when "01010" =>
--   if sprite_R_rlf /= "00000" then
    pixel_R <= sprite_R_rlf;
--   else
```

```
--   pixel_R <= "11111";
 --  end if;

--   if sprite_G_rlf /= "00000" then
   pixel_G <= sprite_G_rlf;
--   else
--  pixel_G <= "11111";
 --  end if;

 --  if sprite_B_rlf /= "00000" then
   pixel_B <= sprite_B_rlf;
--   else
--   pixel_B <= "11111";
--   end if;

when "01011" =>                          -- rrf
  --  if sprite_R_rrf /= "00000" then
   pixel_R <= sprite_R_rrf;
--   else
--   pixel_R <= "11111";
 --  end if;

 --  if sprite_G_rrf /= "00000" then
   pixel_G <= sprite_G_rrf;
```

```
--   else
--   pixel_G <= "11111";
 --  end if;


 --  if sprite_B_rrf /= "00000" then
   pixel_B <= sprite_B_rrf;
--   else
--   pixel_B <= "11111";
--   end if;


 when "01100" =>                                                    --dead
 pixel_R <= sprite_R_dead;
 pixel_G <= sprite_G_dead;
 pixel_B <= sprite_B_dead;



 when others =>
 pixel_R <= "11111";
 pixel_G <= "11111";
 pixel_B <= "11111";
          end case;
```

```
end if;
end process;

end rtl;
```

guard_controller

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity guard_controller is
  port(
clk   : in std_logic;
address: in unsigned (4 downto 0);
vertical: in integer;
horizontal: in integer;

pixel_R: out std_logic_vector (4 downto 0);
pixel_G: out std_logic_vector (4 downto 0);
pixel_B: out std_logic_vector (4 downto 0)
  );
```

```
end guard_controller;

architecture rtl of guard_controller is

------------------------------------------------------------sprite_down----------------------------

component guard_red_down

  port(

clk   : in std_logic;

addr  : in integer;

data  : out std_logic_vector (4 downto 0)

  );

end component;

component guard_green_down

  port(

clk   : in std_logic;

addr  : in integer;

data  : out std_logic_vector (4 downto 0)

  );

end component;

component guard_blue_down

  port(

clk   : in std_logic;
```

```
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_dlf-------------------------------
component guard_red_dlf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_green_dlf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_blue_dlf
 port(
clk   : in std_logic;
```

```
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_drf-------------------------
component guard_red_drf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_green_drf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_blue_drf
 port(
clk   : in std_logic;
```

```
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

------------------------------------------------------------sprite_up----------------------------
component guard_red_up
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_green_up
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_blue_up
 port(
```

```
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_ulf-------------------------------
component guard_red_ulf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_green_ulf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_blue_ulf
 port(
```

```
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;
------------------------------------------------------------sprite_urf-------------------------
component guard_red_urf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_green_urf
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_blue_urf
 port(
```

```
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

------------------------------------------------------------sprite_left----------------------------
component guard_red_left
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_green_left
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
 );
end component;

component guard_blue_left
```

```
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
------------------------------------------------------------sprite_llf--------------------------------
component guard_red_llf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component guard_green_llf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component guard_blue_llf
```

```
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
------------------------------------------------------------sprite_lrf-------------------------
component guard_red_lrf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component guard_green_lrf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component guard_blue_lrf
```

```
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

------------------------------------------------------------sprite_right----------------------------
component guard_red_right
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component guard_green_right
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
```

```
component guard_blue_right
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
-------------------------------------------------------------sprite_rlf-------------------------------
component guard_red_rlf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component guard_green_rlf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
```

```
component guard_blue_rlf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
------------------------------------------------------------sprite_rrf-------------------------
component guard_red_rrf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;

component guard_green_rrf
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic_vector (4 downto 0)
  );
end component;
```

```
component guard_blue_rrf

  port(

clk   : in std_logic;

addr  : in integer;

data  : out std_logic_vector (4 downto 0)

  );

end component;




signal ver : integer;

signal hor : integer;

signal monster_R_down,monster_G_down,monster_B_down : std_logic_vector(4
downto 0);

signal monster_R_dlf,monster_G_dlf,monster_B_dlf : std_logic_vector(4 downto 0);

signal monster_R_drf,monster_G_drf,monster_B_drf : std_logic_vector(4 downto 0);


signal monster_R_up,monster_G_up,monster_B_up : std_logic_vector(4 downto 0);

signal monster_R_ulf,monster_G_ulf,monster_B_ulf : std_logic_vector(4 downto 0);

signal monster_R_urf,monster_G_urf,monster_B_urf : std_logic_vector(4 downto 0);




signal monster_R_left,monster_G_left,monster_B_left : std_logic_vector(4 downto
0);

signal monster_R_llf,monster_G_llf,monster_B_llf : std_logic_vector(4 downto 0);

signal monster_R_lrf,monster_G_lrf,monster_B_lrf : std_logic_vector(4 downto 0);
```

```
signal monster_R_right,monster_G_right,monster_B_right : std_logic_vector(4
downto 0);

signal monster_R_rlf,monster_G_rlf,monster_B_rlf : std_logic_vector(4 downto 0);

signal monster_R_rrf,monster_G_rrf,monster_B_rrf : std_logic_vector(4 downto 0);


begin

----------------------------------------down ----------------------------------

guard_R_down: guard_red_down port map (clk,ver*32+hor,monster_R_down);

guard_G_down: guard_green_down port map (clk,ver*32+hor,monster_G_down);

guard_B_down: guard_blue_down port map (clk,ver*32+hor,monster_B_down);


guard_R_dlf: guard_red_dlf port map (clk,ver*32+hor,monster_R_dlf);

guard_G_dlf: guard_green_dlf port map (clk,ver*32+hor,monster_G_dlf);

guard_B_dlf: guard_blue_dlf port map (clk,ver*32+hor,monster_B_dlf);


guard_R_drf: guard_red_drf port map (clk,ver*32+hor,monster_R_drf);

guard_G_drf: guard_green_drf port map (clk,ver*32+hor,monster_G_drf);

guard_B_drf: guard_blue_drf port map (clk,ver*32+hor,monster_B_drf);


----------------------------------------------up---------------------------------

guard_R_up: guard_red_up port map (clk,ver*32+hor,monster_R_up);

guard_G_up: guard_green_up port map (clk,ver*32+hor,monster_G_up);

guard_B_up: guard_blue_up port map (clk,ver*32+hor,monster_B_up);
```

```
guard_R_ulf: guard_red_ulf port map (clk,ver*32+hor,monster_R_ulf);
guard_G_ulf: guard_green_ulf port map (clk,ver*32+hor,monster_G_ulf);
guard_B_ulf: guard_blue_ulf port map (clk,ver*32+hor,monster_B_ulf);

guard_R_urf: guard_red_urf port map (clk,ver*32+hor,monster_R_urf);
guard_G_urf: guard_green_urf port map (clk,ver*32+hor,monster_G_urf);
guard_B_urf: guard_blue_urf port map (clk,ver*32+hor,monster_B_urf);

----------------------------------------------left-------------------------------
guard_R_left: guard_red_left port map (clk,ver*32+hor,monster_R_left);
guard_G_left: guard_green_left port map (clk,ver*32+hor,monster_G_left);
guard_B_left: guard_blue_left port map (clk,ver*32+hor,monster_B_left);

guard_R_llf: guard_red_llf port map (clk,ver*32+hor,monster_R_llf);
guard_G_llf: guard_green_llf port map (clk,ver*32+hor,monster_G_llf);
guard_B_llf: guard_blue_llf port map (clk,ver*32+hor,monster_B_llf);

guard_R_lrf: guard_red_lrf port map (clk,ver*32+hor,monster_R_lrf);
guard_G_lrf: guard_green_lrf port map (clk,ver*32+hor,monster_G_lrf);
guard_B_lrf: guard_blue_lrf port map (clk,ver*32+hor,monster_B_lrf);

----------------------------------------------right-----------------------------
guard_R_right: guard_red_right port map (clk,ver*32+hor,monster_R_right);
guard_G_right: guard_green_right port map (clk,ver*32+hor,monster_G_right);
```

```
guard_B_right: guard_blue_right port map (clk,ver*32+hor,monster_B_right);


guard_R_rlf: guard_red_rlf port map (clk,ver*32+hor,monster_R_rlf);
guard_G_rlf: guard_green_rlf port map (clk,ver*32+hor,monster_G_rlf);
guard_B_rlf: guard_blue_rlf port map (clk,ver*32+hor,monster_B_rlf);


guard_R_rrf: guard_red_rrf port map (clk,ver*32+hor,monster_R_rrf);
guard_G_rrf: guard_green_rrf port map (clk,ver*32+hor,monster_G_rrf);
guard_B_rrf: guard_blue_rrf port map (clk,ver*32+hor,monster_B_rrf);


process(clk)
begin
if rising_edge(clk) then
  hor <= horizontal;
  ver <= vertical;
end if;
end process;


process(clk)
begin
if rising_edge(clk) then
case address is
---------------------------------------down---------------------------------------
 when "00000" =>                                                    -- down
```

```
--    if monster_R_down /= "00000" then
      pixel_R <= monster_R_down;
--    else
--    pixel_R <= "11111";
--    end if;

 --   if monster_G_down /= "00000" then
      pixel_G <= monster_G_down;
--    else
--    pixel_G <= "11111";
--    end if;



      pixel_B <= monster_B_down;

    when "00001" =>                                          -- dlf

      pixel_R <= monster_R_dlf;



      pixel_G <= monster_G_dlf;
```

```
   pixel_B <= monster_B_dlf;

when "00010" =>                         -- drf

   pixel_R <= monster_R_drf;



   pixel_G <= monster_G_drf;

   pixel_B <= monster_B_drf;

-----------------------------------------------------------------up----------------------------
 when "00011" =>

   pixel_R <= monster_R_up;

   pixel_G <= monster_G_up;

   pixel_B <= monster_B_up;



 when "00100" =>
```

```
   pixel_R <= monster_R_ulf;




   pixel_G <= monster_G_ulf;




   pixel_B <= monster_B_ulf;



 when "00101" =>                        -- urf

   pixel_R <= monster_R_urf;

   pixel_G <= monster_G_urf;

   pixel_B <= monster_B_urf;

-------------------------------------------------------------left---------------------------

 when "00110" =>

   pixel_R <= monster_R_left;

   pixel_G <= monster_G_left;
```

```
   pixel_B <= monster_B_left;



 when "00111" =>

   pixel_R <= monster_R_llf;

   pixel_G <= monster_G_llf;

   pixel_B <= monster_B_llf;



 when "01000" =>                        -- lrf

   pixel_R <= monster_R_lrf;

   pixel_G <= monster_G_lrf;

   pixel_B <= monster_B_lrf;

-------------------------------------------------------------------right----------------------------

 when "01001" =>
```

```
    pixel_R <= monster_R_right;

    pixel_G <= monster_G_right;

    pixel_B <= monster_B_right;



  when "01010" =>

    pixel_R <= monster_R_rlf;

    pixel_G <= monster_G_rlf;

    pixel_B <= monster_B_rlf;



 when "01011" =>                    -- rrf

    pixel_R <= monster_R_rrf;

    pixel_G <= monster_G_rrf;

    pixel_B <= monster_B_rrf;
```

```
  when others =>
  pixel_R <= "11111";
  pixel_G <= "11111";
  pixel_B <= "11111";
end case;

end if;
end process;

end rtl;
```

info_controller

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity info_controller is
  port(
clk   : in std_logic;
address: integer;
```

```
vertical: in integer;
horizontal: in integer;

pixel_R: out std_logic_vector(4 downto 0);
pixel_G: out std_logic_vector(4 downto 0);
pixel_B: out std_logic_vector(4 downto 0)
  );

end info_controller;

architecture rtl of info_controller is
----------------------------------COMPONENTS-------------------------------------------
component num0
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;
component num1
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
```

```
  );
end component;
component num2
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;
component num3
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;
component num4
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;
component num5
```

```
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;
component num6
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;
component num7
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;
component num8
  port(
clk   : in std_logic;
addr  : in integer;
```

```
data  : out std_logic
 );
end component;
component num9
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
 );
end component;
component numl
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
 );
end component;
component numv
 port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
 );
end component;
```

```
component numx
  port(
clk   : in std_logic;
addr  : in integer;
data  : out std_logic
  );
end component;

----------------------------------SIGNALS-------------------------------------------------------
signal ver : integer;
signal hor : integer;
signal sum : integer;
signal out0: std_logic;
signal out1: std_logic;
signal out2: std_logic;
signal out3: std_logic;
signal out4: std_logic;
signal out5: std_logic;
signal out6: std_logic;
signal out7: std_logic;
signal out8: std_logic;
signal out9: std_logic;
signal outl: std_logic;
signal outv: std_logic;
```

```
signal outx: std_logic;



begin
pout0: num0 port map(clk,ver*32+hor,out0);
pout1: num1 port map(clk,ver*32+hor,out1);
pout2: num2 port map(clk,ver*32+hor,out2);
pout3: num3 port map(clk,ver*32+hor,out3);
pout4: num4 port map(clk,ver*32+hor,out4);
pout5: num5 port map(clk,ver*32+hor,out5);
pout6: num6 port map(clk,ver*32+hor,out6);
pout7: num7 port map(clk,ver*32+hor,out7);
pout8: num8 port map(clk,ver*32+hor,out8);
pout9: num9 port map(clk,ver*32+hor,out9);
poutl: numl port map(clk,ver*32+hor,outl);
poutv: numv port map(clk,ver*32+hor,outv);
poutx: numx port map(clk,ver*32+hor,outx);
process(clk)
begin
if rising_edge(clk) then
  hor <= horizontal;
  ver <= vertical;
end if;
end process;
```

```
process(clk)
begin
 if rising_edge(clk) then
 case address is
 when 0 => if out0 ='1' then
              pixel_R <= "00000";
              pixel_G <= "00000";
              pixel_B <= "00000";
           else
              pixel_R <= "11111";
              pixel_G <= "11111";
              pixel_B <= "11111";
           end if;
 when 1 => if out1 ='1' then
              pixel_R <= "00000";
              pixel_G <= "00000";
              pixel_B <= "00000";
           else
              pixel_R <= "11111";
              pixel_G <= "11111";
              pixel_B <= "11111";
           end if;
 when 2 => if out2 ='1' then
```

```
            pixel_R <= "00000";
            pixel_G <= "00000";
            pixel_B <= "00000";
          else
            pixel_R <= "11111";
            pixel_G <= "11111";
            pixel_B <= "11111";
          end if;
when 3 => if out3 ='1' then
            pixel_R <= "00000";
            pixel_G <= "00000";
            pixel_B <= "00000";
          else
            pixel_R <= "11111";
            pixel_G <= "11111";
            pixel_B <= "11111";
          end if;
when 4 => if out4 ='1' then
            pixel_R <= "00000";
            pixel_G <= "00000";
            pixel_B <= "00000";
          else
            pixel_R <= "11111";
            pixel_G <= "11111";
```

```
            pixel_B <= "11111";
        end if;
when 5 => if out5 ='1' then
            pixel_R <= "00000";
            pixel_G <= "00000";
            pixel_B <= "00000";
        else
            pixel_R <= "11111";
            pixel_G <= "11111";
            pixel_B <= "11111";
        end if;
when 6 => if out6 ='1' then
            pixel_R <= "00000";
            pixel_G <= "00000";
            pixel_B <= "00000";
        else
            pixel_R <= "11111";
            pixel_G <= "11111";
            pixel_B <= "11111";
        end if;
when 7 => if out7 ='1' then
            pixel_R <= "00000";
            pixel_G <= "00000";
            pixel_B <= "00000";
```

```
        else
          pixel_R <= "11111";
          pixel_G <= "11111";
          pixel_B <= "11111";
        end if;
when 8 => if out8 ='1' then
          pixel_R <= "00000";
          pixel_G <= "00000";
          pixel_B <= "00000";
        else
          pixel_R <= "11111";
          pixel_G <= "11111";
          pixel_B <= "11111";
        end if;
when 9 => if out9 ='1' then
          pixel_R <= "00000";
          pixel_G <= "00000";
          pixel_B <= "00000";
        else
          pixel_R <= "11111";
          pixel_G <= "11111";
          pixel_B <= "11111";
        end if;
when 10 => if outx ='1' then      ------XXXXX
```

```
            pixel_R <= "00000";
            pixel_G <= "00000";
            pixel_B <= "00000";
          else
            pixel_R <= "11111";
            pixel_G <= "11111";
            pixel_B <= "11111";
          end if;
when 11 => if outl ='1' then
            pixel_R <= "00000";   -- LLL
            pixel_G <= "00000";
            pixel_B <= "00000";
          else
            pixel_R <= "11111";
            pixel_G <= "11111";
            pixel_B <= "11111";
          end if;
when 12 => if outv ='1' then
            pixel_R <= "00000";    --------- VVVVV
            pixel_G <= "00000";
            pixel_B <= "00000";
          else
            pixel_R <= "11111";
            pixel_G <= "11111";
```

```
            pixel_B <= "11111";
          end if;
when others =>   pixel_R <= "11111";
               pixel_G <= "11111";
               pixel_B <= "11111";

 end case;

 end if;

end process;

end rtl;
```

hello_world.c

```
#include <io.h>
#include <system.h>
#include <stdio.h>
#include <sys/alt_irq.h>
#include <alt_types.h>
//#include "basic_io.h"
//#include <alt_types.h>
//#include "alt_up_ps2_port.h"
```

```
//#include "ps2_keyboard.h"
//#include "ps2_mouse.h"
#define IOWR_VGA_DATA(base, offset, data) \
  IOWR_16DIRECT(base, (offset) * 2, data)
#define IORD_VGA_DATA(base, offset) \
  IORD_16DIRECT(base, (offset) * 2)
#define IOWR_WM8731_DATA(base, offset, data) \
  IOWR_16DIRECT(base, (offset) * 2, data)
void gbreakice(int gbx,int gby);



int sprite_mode = 0;        //for sprite direction
int gSTEP=32;
int gx=448,gy=32,gflag= 0;
int guardflag = 0;
int x,y;
int gsprite_mode = 0;
int gtile_h,gtile_v;
int gamelevel=0,gamemode=0;//0 start 1 dead
int grecord=0,brecord=0;
int record[3]={22,30,16}; //22   30  16
int diff_x,diff_y,fruit_flag=0, move_flag = 0,first_y = 0, x_priority = 0, y_priority = 0;
int gspeed_flag  = 0, gspeed_counter = 0;
```

```
int g_normal = 0, eat_flag = 0; normal_counter = 0, normal_direction = 0;

int guard_table_index = 0;

int guard_table[30] = {0,3,2,1,2,0,1,1,0,3,2,1,3,2,0,3,2,2,2,1,3,0,0,2,3,1,1,2,3,0};

int guard_lock_flag = 0;    //ensure that the ice won't affect guard tile

int guard_lock_h = 0, guard_lock_v = 0;

int guard_reset_flag = 0, guard_reset_counter = 0;

int gameflag1 = 0, gameflag2 = 0;


void gmove_up();

void gmove_down();

void gmove_right();

void gmove_left();


int abs(int x);

void gameover();

void gamewin();

int dealy();

void reset();


//int mapping[3][15][20]={

//{{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

//{0,0,0,0,0,0,0,0,3,3,3,3,0,0,0,1,0,0,0,0},

//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0},

//{0,0,0,0,7,7,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
```

```
//{0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,1,0,0,0,0},
//{0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,1,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{0,0,0,0,0,0,0,0,2,2,2,2,0,0,0,0,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,0,0,3,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1},
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
//
//
//
//{{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0},
//{0,0,0,0,2,2,0,0,0,0,0,0,0,0,0,1,1,1,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,1,1,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,1,1,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0},
//{1,0,3,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0},
```

```
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,0,0,7,7,7,1,1,0,0,0,0,0,0,0,0,3,3,0,0},
//{1,0,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1},
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
//
//{{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{0,0,0,0,0,0,0,0,3,3,3,3,0,0,0,1,0,0,0,0},
//{0,0,0,0,0,0,0,0,7,7,7,7,0,0,0,1,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
//{0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
//{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{0,0,0,0,0,0,0,0,2,2,0,2,2,0,0,0,0,0,0,0},
//{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7},
//{1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,7,0},IOWR_VGA_DATA(VGA_BASE, 15, 0);


//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,0,0,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
//{1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1},
//{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}}
//
//};
```

```
int mapping[3][15][20]={
{{7,4,4,4,4,7,7,7,4,4,4,4,7,7,7,4,4,4,4,7},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,1,1,3,1,1,0,1,1,3,1,1},
{0,0,0,2,0,0,7,7,0,1,1,3,1,1,0,1,1,3,1,1},
{0,0,0,2,0,7,7,7,7,1,1,3,1,1,0,1,1,3,1,1},
{0,0,0,0,0,0,0,0,0,1,1,3,1,1,0,1,1,3,1,1},
{0,0,3,3,0,0,0,0,0,1,1,1,1,1,0,1,1,1,1,1},
{7,7,7,0,0,0,0,0,0,1,1,1,1,1,0,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
{1,1,2,2,2,1,1,1,2,2,2,2,1,1,1,2,2,2,1,1},
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
{{7,4,4,4,4,7,7,7,4,4,4,4,7,7,7,4,4,4,4,7},
{0,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1},
{0,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1},
{1,1,7,7,0,0,0,2,0,0,0,0,2,0,0,0,7,7,1,1},
```

{1,1,7,0,0,0,3,1,3,0,0,3,1,3,0,0,0,7,1,1},

{1,1,0,0,0,0,0,2,0,0,0,0,2,0,0,0,0,0,1,1},

{1,1,0,0,0,0,3,1,3,0,0,3,1,3,0,0,0,0,1,1},

{1,1,0,0,0,0,0,2,0,0,0,0,2,0,0,0,0,0,1,1},

{1,1,0,0,0,0,3,1,3,0,0,3,1,3,0,0,0,0,1,1},

{1,1,0,0,0,0,0,2,0,0,0,0,2,0,0,0,0,0,1,1},

{1,1,0,0,0,0,3,1,3,0,0,3,1,3,0,0,0,0,1,1},

{1,1,7,0,0,0,0,2,0,0,0,0,2,0,0,0,0,7,1,1},

{1,1,7,7,0,0,3,1,3,0,0,3,1,3,0,0,7,7,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}},

{{7,4,4,4,4,7,7,7,4,4,4,4,7,7,7,4,4,4,4,7},

{0,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1},

{0,0,1,1,7,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1},

{1,1,1,7,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1},

{1,1,7,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,3,1,1},

{1,1,1,1,1,2,0,2,0,2,0,2,1,1,0,1,1,0,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,1,1},

{1,1,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},

{1,1,1,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},

{1,1,1,1,3,1,1,1,1,1,1,1,1,1,1,1,1,3,1,1},

{1,1,1,1,1,3,1,1,1,1,1,1,1,1,1,1,1,0,1,1},

```
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,1,1},

{1,2,0,2,0,2,0,2,1,1,7,7,1,1,1,1,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}}


};



 ////////////////////Interupt Handler/////////////////////////////////////////////

static void irqhandler (void * context, alt_u32 id)

{


   if(guard_reset_flag == 1){

      if(guard_reset_counter == 20){

       guard_reset_counter = 0;

       guard_reset_flag = 0;

      }else{

       guard_reset_counter ++;

       return;

      }

   }


   if(gamemode == 2)

     return;
```

```
gtile_h = gx/32;
gtile_v = gy/32;

diff_x = gx - x;
diff_y = gy - y;
move_flag = 0;            //reset move flag
guard_lock_h = 0;         //clear tile offset for lock
guard_lock_v = 0;

if(gamemode == 0 && eat_flag == 0){

 if(fruit_flag != 3 && normal_direction == 0){

   fruit_flag = 0;
   gmove_left();
   move_flag = 1;
 }

else if(fruit_flag != 1 && move_flag == 0 && normal_direction == 1){
        //     printf("up\n");
             fruit_flag = 0;
             gmove_up();
             move_flag = 1;
      }
```

```
  else if(fruit_flag != 4 && move_flag == 0 && normal_direction == 2){
//     printf("right\n");
      fruit_flag = 0;
      gmove_right();
      move_flag = 1;
   }



   else if(fruit_flag != 2 && move_flag == 0 && normal_direction == 3){
            //  printf("down\n");
              fruit_flag = 0;
              gmove_down();
              move_flag = 1;
    }


    diff_x = gx - x;          //calculate again
    diff_y = gy - y;


    if(abs(diff_y) < 32  && abs(diff_x) < 32){
       gamemode=1;
       IOWR_VGA_DATA(VGA_BASE, 14, 0);
       gameover();
    }
```

```
  normal_counter ++;
 if(normal_counter == 5){
   normal_counter = 0;
 if(guard_table_index != 29)
  guard_table_index ++;
 else
  guard_table_index = 0;

  normal_direction = guard_table[guard_table_index];
 }

  IOWR_VGA_DATA(VGA_BASE, 30, 0);      //movement down
  IOWR_16DIRECT(IRQSOURCE_BASE, 2, 0);   //speed down
}
else if(gamemode == 0 && eat_flag == 1 )
{
    //   if(diff_x!=0){
    IOWR_16DIRECT(IRQSOURCE_BASE, 2, 1);
    if(first_y != 1){
          if(diff_x>0){
             if(fruit_flag != 3) {
             fruit_flag = 0;
             gmove_left();
```

```
            move_flag = 1;
            }
            }
        if(diff_x<0){
            if(fruit_flag != 4 && move_flag == 0){
            fruit_flag = 0;guard_lock_flag = 1;  // ice lock up
            gmove_right();
            move_flag = 1;
            }
            }guard_lock_flag = 1;  // ice lock up
}
        if(diff_y>0){
            if(fruit_flag != 1 && move_flag == 0){
            fruit_flag = 0;
            gmove_up();
            move_flag = 1;
            first_y = 0;
            }
            }
        if(diff_y<0) {
            if(fruit_flag != 2 && move_flag == 0){
            fruit_flag = 0;
            gmove_down();
            move_flag = 1;
```

```
            first_y = 0;
          }
        }
//-------------------------------------------------when there are fruit  on the way----
        if(diff_x == 0 && move_flag == 0){
         if(x_priority == 0){
            x_priority = 1;
           if(fruit_flag != 3 && move_flag == 0 && gx > 32 ) {
            fruit_flag = 0;
            gmove_left();
            move_flag = 1;
            first_y = 1;
            }
         }else{
            x_priority = 0;
           if(fruit_flag != 4 && move_flag == 0 && gx < 608){
            fruit_flag = 0;
            gmove_right();
            move_flag = 1;
            first_y = 1;
            }
         }
        }
```

```
 if(diff_y == 0 && move_flag == 0){
  if(y_priority == 0){
     y_priority = 1;

    if(fruit_flag != 1 && move_flag == 0 && gy > 32 ){
//     printf("up\n");
     fruit_flag = 0;
     gmove_up();
     move_flag = 1;
     }
  }else{
     y_priority = 0;

     if(fruit_flag != 2 && move_flag == 0 && gy <448 ){
   //      printf("down\n");
     fruit_flag = 0;
     gmove_down();
     move_flag = 1;
     }
  }
 }

diff_x = gx - x;          //calculate again
diff_y = gy - y;
```

```
gtile_v = gy/32;
        if(abs(diff_y) < 32  && abs(diff_x) < 32){
           gamemode=1;
           IOWR_VGA_DATA(VGA_BASE, 14, 0);
           gameover();
           }


 /// printf("gx=%d  ",gx);
 // printf("gy=%d  \n",gy);


  gspeed_counter ++;
  if(gspeed_counter == 20){                //change speed mode
  gspeed_counter = 0;
  eat_flag = 0;
  IOWR_VGA_DATA(VGA_BASE, 30, 0);     //movement down
  IOWR_16DIRECT(IRQSOURCE_BASE, 2, 0);
  }
 /*  if(gspeed_counter == 10 && gspeed_flag =diff_y==0 && diff_x==0=
0){                //change speed mode
  gspeed_counter = 0;
  gspeed_flag = 1;
  IOWR_VGA_DATA(VGA_BASE, 31, 0);    // movement up
  }
```

```
   if(gspeed_counter == 20 && gspeed_flag == 1){
     gspeed_counter = 0;
     gspeed_flag = 0;
     IOWR_VGA_DATA(VGA_BASE, 30, 0);   // movement down
   }

     if(gspeed_flag == 0){
     IOWR_16DIRECT(IRQSOURCE_BASE, 2, 0);  // speed down

     }else{
     IOWR_16DIRECT(IRQSOURCE_BASE, 2, 1);  // speed up

     }*/
   //   printf ("READ DATA :%d\n", IORD_16DIRECT(IRQSOURCE_BASE, 2));
   //   printf("%d\n",gspeed_flag);
   //   printf("%d\n",gspeed_counter);
  }
}
 ////////////////////Main
Function/////////////////////////////////////////////////
int main()
{
```

```
int tile_h,tile_v;          // the position for the tile unit

unsigned char code;

x = 0;

y = 32;



int key_enable = 1;

printf("Ready to start\n");

IOWR_VGA_DATA(VGA_BASE, 30, 0); // setup speed

IOWR_VGA_DATA(VGA_BASE, 0, x);// set initial position of the Ice cream

IOWR_VGA_DATA(VGA_BASE, 1, y);

IOWR_VGA_DATA(VGA_BASE, 9, gy);// set initial position of the guard

IOWR_VGA_DATA(VGA_BASE, 8, gx); //


void Ice_gen( int tile_v,int tile_h,int sprite_mode);

void grapeNbanana(int x,int y);


alt_irq_register( IRQSOURCE_IRQ, NULL,

        (void*)irqhandler );// register IRQSOURCE



while(gamemode == 0){

 while (!IORD_8DIRECT(DE2_PS2_0_BASE,0));
```

```
code = IORD_8DIRECT(DE2_PS2_0_BASE, 4);


tile_h = x/32;

tile_v = y/32;


printf("x %d y %d\n", x, y);

printf("%d \n",sprite_mode);


if(gameflag1 == 1){

  x = 0;

  y = 32;

  gameflag1 = 0;

}


if(gameflag2 == 1){

    x = 0;

    y = 32;

    gameflag2 = 0;

}




switch(code){


  case 0xf0:
```

```
key_enable = 0;

break;


case 0x1b:

if(gamemode == 0){

diff_x = gx - x;           //calculate again

diff_y = gy - y;

if (abs(diff_y) < 32  && abs(diff_x) < 32) {

    gamemode=1;

    IOWR_VGA_DATA(VGA_BASE, 14, 0);

    gameover();

   }


 if(key_enable == 1){IOWR_VGA_DATA(VGA_BASE, 0, x);// set initial position of the
Ice cream

IOWR_VGA_DATA(VGA_BASE, 1, y);

 if (sprite_mode!=0){              //change direction

    IOWR_VGA_DATA(VGA_BASE, 1, y);

    sprite_mode=0;

    }else{

  if( y < 448 && mapping[gamelevel][tile_v +1][tile_h] != 1){


       y = y + 16;

   if (abs(diff_y) < 32  && abs(diff_x) < 32) {
```

```
    gamemode=1;
    IOWR_VGA_DATA(VGA_BASE, 14, 0);
    gameover();    x = 0;
y = 32;
   }else {
    IOWR_VGA_DATA(VGA_BASE, 1, y);

    while(IORD_VGA_DATA(VGA_BASE,1) != 15) ;
    while(IORD_VGA_DATA(VGA_BASE,1) != 0)
    IOWR_VGA_DATA(VGA_BASE, 18, 0);
    grapeNbanana(tile_h,tile_v+1);
    y = y + 16;
   }
  sprite_mode = 0;

 }
 else{
    sprite_mode = 0;
    IOWR_VGA_DATA(VGA_BASE, 1, y);

    }
 }

key_enable=0; //down=0
```

```
    }
    else {

     key_enable = 1;
    }
    }
     break;
//
    case 0x1d:
    if(gamemode == 0){
    diff_x = gx - x;          //calculate again
    diff_y = gy - y;
    if (abs(diff_y) < 32  && abs(diff_x) < 32) {
       gamemode=1;
       IOWR_VGA_DATA(VGA_BASE, 14, 0);
       gameover();
      }

    if(key_enable == 1){
    if (sprite_mode!=1){
       IOWR_VGA_DATA(VGA_BASE, 3, y);
       sprite_mode=1;
       }
    else{
```

```
if( y > 32 && mapping[gamelevel][tile_v-1][tile_h] != 1){


  y = y - 16;
  if (abs(diff_y) < 32  && abs(diff_x) < 32){
    gamemode=1;
    IOWR_VGA_DATA(VGA_BASE, 14, 0);
    gameover();
  }else{
 IOWR_VGA_DATA(VGA_BASE, 3, y);
  }
 sprite_mode = 1;
 while(IORD_VGA_DATA(VGA_BASE,1) != 15) ;
while(IORD_VGA_DATA(VGA_BASE,1) != 0)// set initial position of the Ice cream
IOWR_VGA_DATA(VGA_BASE, 18, 0);
y = y -16;
grapeNbanana(tile_h,tile_v-1);
}
else {sprite_mode = 1;
 IOWR_VGA_DATA(VGA_BASE, 3, y);
 }
}


 key_enable=0; // up=1
}
```

```
else
  key_enable = 1;
}
break;

case 0x1c:
if(gamemode == 0){
diff_x = gx - x;          //calculate again
diff_y = gy - y;
if (abs(diff_y) < 32  && abs(diff_x) < 32) {
   gamemode=1;
   IOWR_VGA_DATA(VGA_BASE, 14, 0);
   gameover();
  }

if(key_enable == 1){
if (sprite_mode!=2){
   IOWR_VGA_DATA(VGA_BASE, 0, x);
   sprite_mode=2;
   }

else{
if( x > 0 && mapping[gamelevel][tile_v  ][tile_h-1] != 1){
```

```
   x = x - 16;
  if (abs(diff_y) < 32  && abs(diff_x) < 32){
   gamemode=1;
   IOWR_VGA_DATA(VGA_BASE, 14, 0);
   gameover();
   }
  else {
  IOWR_VGA_DATA(VGA_BASE, 0, x);

  }
sprite_mode = 2;
while(IORD_VGA_DATA(VGA_BASE,1) != 15) ;
while(IORD_VGA_DATA(VGA_BASE,1) != 0)
IOWR_VGA_DATA(VGA_BASE, 18, 0);
x = x - 16;
grapeNbanana(tile_h-1,tile_v);
}
else{
   sprite_mode = 2;
IOWR_VGA_DATA(VGA_BASE, 0, x);
}
}
```

```
 key_enable=0;  //left=2


}
  else
  key_enable = 1;
}
  break;


case 0x23:
if(gamemode == 0){
diff_x = gx - x;           //calculate again
diff_y = gy - y;
if (abs(diff_y) < 32  && abs(diff_x) < 32) {
    gamemode=1;
    IOWR_VGA_DATA(VGA_BASE, 14, 0);
    gameover();
   }


 if(key_enable == 1){
 if (sprite_mode!=3){
    IOWR_VGA_DATA(VGA_BASE, 2, x);
    sprite_mode=3;
```

```
    }
 else{
if( x < 608 && mapping[gamelevel][tile_v  ][tile_h + 1] != 1 ){


   x = x + 16;
   if (abs(diff_y) < 32  && abs(diff_x) < 32){
    gamemode=1;
    IOWR_VGA_DATA(VGA_BASE, 14, 0);
    gameover();
    }
   else {
    IOWR_VGA_DATA(VGA_BASE, 2, x);


 }
sprite_mode = 3;
while(IORD_VGA_DATA(VGA_BASE,1) != 15) ;
while(IORD_VGA_DATA(VGA_BASE,1) != 0)
    IOWR_VGA_DATA(VGA_BASE, 18, 0);
  x = x + 16;
  grapeNbanana(tile_h+1,tile_v);
}
else{
    sprite_mode = 3;
    IOWR_VGA_DATA(VGA_BASE, 2, x);
```

```
    }
    }



     key_enable=0; //right=3
     }
     else
     key_enable = 1;
    }

     break;
//
    case 0x29:
    if(gamemode == 0){
    if(key_enable==1){
    Ice_gen(tile_v,tile_h,sprite_mode);
    key_enable=0;
    }
   else
       key_enable=1;
    }
   break;
```

```
  case 0x5A:
  if(key_enable == 1){
    reset();
  }
  else
      key_enable=1;
  break;



 }
 printf("System terminated normally");
 }
 printf("System unnormal\n");
}
 ////////////////////////////////grape and banana map
change//////////////////////////////
 void grapeNbanana(int xx,int yy){
   if (mapping[gamelevel][yy][xx]==2) {        // 2 banana
           mapping[gamelevel][yy][xx]=0;
           IOWR_VGA_DATA(VGA_BASE, 12, yy);   // banana x,y
           IOWR_VGA_DATA(VGA_BASE, 13, xx);
           IOWR_VGA_DATA(VGA_BASE, 17, xx);  // banana record
           IOWR_WM8731_DATA(WM8731_BASE,31,1); //sound
           brecord ++;
           record[gamelevel]--;
```

```
        printf("banana_record=%d\n ",brecord);
        eat_flag = 1;
        if(record[2] == 0){
          gamewin();
      //    IOWR_VGA_DATA(VGA_BASE,29,0);//address'00100'  reset
          return;
        }
        }
    else if (mapping[gamelevel][yy][xx]==3) {      // 3 grape
        mapping[gamelevel][yy][xx]=0;
        IOWR_VGA_DATA(VGA_BASE, 6, yy);
        IOWR_VGA_DATA(VGA_BASE, 7, xx);
        IOWR_VGA_DATA(VGA_BASE, 16, xx);  // grape record
        IOWR_WM8731_DATA(WM8731_BASE,31,1);
        grecord ++;
        record[gamelevel]--;
        printf("grape_record=%d\n",grecord);
        eat_flag = 1;
        if(record[2] == 0){
          gamewin();
        //  IOWR_VGA_DATA(VGA_BASE,29,0);//address'00100'
          return;
        }
        }
```

```
if(record[gamelevel]==0) {
  int counter = 0, mm = 0;
  gamelevel++;
  IOWR_VGA_DATA(VGA_BASE, 15, 0);
   guard_reset_flag = 1;
        gx=448,gy=32,gflag= 0;  //reset guard


        sprite_mode = 0;
        for(counter; counter < 50; counter ++)
        {
         for(mm = 0; mm < 20000; mm ++);
         IOWR_VGA_DATA(VGA_BASE,29,0);
         x = 0,y = 32;
         printf("x %d\n,y %d\n",x,y);
         IOWR_VGA_DATA(VGA_BASE, 0, x);// set initial position of the Ice cream
         IOWR_VGA_DATA(VGA_BASE, 1, y);
         }//address'00100'   // reset several times
  printf("gamelevel=%d ",gamelevel);


 if(gamelevel == 1){
    gameflag1 = 1;
 }
 if(gamelevel == 2){
    gameflag2 = 1;
```

```
      }

       };



 }

 ////////////////////////////////guard
movement/////////////////////////////////////////////

  void gmove_up(){

//   int i,j;

  if( gy > 32 && mapping[gamelevel][gtile_v-1][gtile_h] !=1){



     if(mapping[gamelevel][gtile_v-1][gtile_h] <4 && mapping[gamelevel][gtile_v-
1][gtile_h] >1){

       fruit_flag=1;



       normal_counter = 0;

     if(guard_table_index != 29)

      guard_table_index ++;

    else

      guard_table_index = 0;



     normal_direction = guard_table[guard_table_index];

     }



     else {

     gy  = gy -32;
```

```
     IOWR_VGA_DATA(VGA_BASE,11, gy);
     guard_lock_flag = 1;  // ice lock up
//    }
    gsprite_mode = 1;
        }  // up=1
   }
  else if( gy > 32  && mapping[gamelevel][gtile_v-1][gtile_h] ==1)
   {IOWR_VGA_DATA(VGA_BASE,11, gy);gsprite_mode=4;
   if(mapping[gamelevel][gtile_v -1][gtile_h] == 1){gbreakice(gx,gy-32);}
    }
  }

 void gmove_down(){
//  int i,j;
   if( gy < 448 && mapping[gamelevel][gtile_v +1][gtile_h] != 1){

    if(mapping[gamelevel][gtile_v+1][gtile_h] <4 &&
mapping[gamelevel][gtile_v+1][gtile_h] > 1){
       fruit_flag=2;                    //2 denote fruit is downside
      normal_counter = 0;
     if(guard_table_index != 29)
      guard_table_index ++;
    else
      guard_table_index = 0;
```

```
    normal_direction = guard_table[guard_table_index];
       }
    else {
    gy = gy +32;
    IOWR_VGA_DATA(VGA_BASE, 9, gy);
    guard_lock_flag = 2;  // ice lock down
//    }
    gsprite_mode = 0;//down=0
       }
   }
  else if( gy < 448 && mapping[gamelevel][gtile_v +1][gtile_h] == 1)
    {IOWR_VGA_DATA(VGA_BASE,9, gy);gsprite_mode=5;
      if(mapping[gamelevel][gtile_v +1][gtile_h] == 1){gbreakice(gx,gy+32);}
      }

 }
void gmove_left(){
 if( gx > 0 && mapping[gamelevel][gtile_v ][gtile_h-1] !=1){
    if(mapping[gamelevel][gtile_v][gtile_h-1] < 4 &&
mapping[gamelevel][gtile_v][gtile_h-1] > 1){
      fruit_flag=3;                    //denot left side fruit
      normal_counter = 0;
    if(guard_table_index != 29)
     guard_table_index ++;
   else
```

```
      guard_table_index = 0;


     normal_direction = guard_table[guard_table_index];
      }
     else {
     gx = gx - 32;
     IOWR_VGA_DATA(VGA_BASE, 8, gx);
     guard_lock_flag = 3;  // ice lock left
//   }
   gsprite_mode = 2; //left=2
         }
   }
  else  if( gx > 0 && mapping[gamelevel][gtile_v ][gtile_h-1] ==1)
  {IOWR_VGA_DATA(VGA_BASE,8, gx);gsprite_mode=6;
   if(mapping[gamelevel][gtile_v][gtile_h-1] == 1){gbreakice(gx-32,gy);}
    }
}


void gmove_right(){


 if( gx < 608 && mapping[gamelevel][gtile_v][gtile_h + 1] !=1){
//    for(i =0; i < gSTEP; i++){
//       for(j = 0; j < 8000; j ++ );
     if(mapping[gamelevel][gtile_v][gtile_h+1] < 4 &&
mapping[gamelevel][gtile_v][gtile_h+1] > 1){
```

```
        fruit_flag=4;                       //denote right side fruit
        normal_counter = 0;
       if(guard_table_index != 29)
        guard_table_index ++;
      else
        guard_table_index = 0;
       normal_direction = guard_table[guard_table_index];
         }
       else {
       gx = gx + 32;
       IOWR_VGA_DATA(VGA_BASE, 10, gx);
       guard_lock_flag = 4;  // ice lock right
//     }
      gsprite_mode = 3; //right=3
       }
      }
   else if( gx < 608 && mapping[gamelevel][gtile_v][gtile_h + 1] ==1)
   {
     IOWR_VGA_DATA(VGA_BASE,10, gx);gsprite_mode=7;
      if(mapping[gamelevel][gtile_v][gtile_h+1] == 1){gbreakice(gx+32,gy);}
       }
}
////////////////////////////////GUARD BREAK ICE
//////////////////////////////////////////
void gbreakice(gbx,gby){
```

```
   gbx=gbx/32;

   gby=gby/32;


  mapping[gamelevel][gby][gbx]=0;

  IOWR_VGA_DATA(VGA_BASE,4,gby);//address'00100'

  IOWR_VGA_DATA(VGA_BASE,5,gbx);//address'00101'

}


////////////////////////////////////Ice_generate//////////////////////////////////
//////

  void Ice_gen(tile_v,tile_h,sprite_mode){

  int ice_tile_h,ice_tile_v,j;

  gtile_h = gx/32;

  gtile_v = gy/32;

    if(sprite_mode == 0){                        // face down

       if (mapping[gamelevel][tile_v +1][tile_h]== 1){

           ice_tile_v=tile_v+1;ice_tile_h=tile_h;

           IOWR_WM8731_DATA(WM8731_BASE,31,2);

           for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==1  && ice_tile_v
<15;ice_tile_v++)

             {

               mapping[gamelevel][ice_tile_v][ice_tile_h]=0;

               IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'

               IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'
```

```
          for(j=0;j<70000;j++);// add delay
        }
                              }
else if (mapping[gamelevel][tile_v +1][tile_h]== 0){
      ice_tile_v=tile_v+1;
      ice_tile_h=tile_h;
      IOWR_WM8731_DATA(WM8731_BASE,31,2);
      switch(guard_lock_flag){
        case 0:
       guard_lock_h = 0;
       guard_lock_v = 0;
         break;

       case 1:  // up
         guard_lock_h = 0;
         guard_lock_v = -1;
         break;

       case 2:  // down
         guard_lock_h = 0;
         guard_lock_v = 1;
         break;

       case 3:  // left
```

```
            guard_lock_h = -1;

            guard_lock_v = 0;

            break;


          case 4:  // right

            guard_lock_h = 1;

            guard_lock_v = 0;

            break;

        }

        for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==0 && ice_tile_v
<15;ice_tile_v++) {

          if((ice_tile_v == gtile_v + guard_lock_v && ice_tile_h== gtile_h +
guard_lock_h) || (ice_tile_v == gtile_v && ice_tile_h== gtile_h))

          break;

          else

           { mapping[gamelevel][ice_tile_v][ice_tile_h]=1;

           IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'

           IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'


           for(j=0;j<70000;j++);// add delay

          }

        }

        }

              }
```

```
   else if (sprite_mode == 1){                    // face up
      if (mapping[gamelevel][tile_v-1][tile_h] == 1){
         IOWR_WM8731_DATA(WM8731_BASE,31,2);
          ice_tile_v=tile_v-1;ice_tile_h=tile_h;
          for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==1 && ice_tile_v
>=0;ice_tile_v--)
            { mapping[gamelevel][ice_tile_v][ice_tile_h]=0;
              IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'
              IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'

              for(j=0;j<70000;j++);// add delay
            }
                              }
      else if (mapping[gamelevel][tile_v-1][tile_h] == 0){
         ice_tile_v=tile_v-1;
         ice_tile_h=tile_h;
         IOWR_WM8731_DATA(WM8731_BASE,31,2);
         switch(guard_lock_flag){
             case 0:
            guard_lock_h = 0;
            guard_lock_v = 0;
              break;

            case 1:  // up
              guard_lock_h = 0;
```

```
            guard_lock_v = -1;
            break;

          case 2:  // down
            guard_lock_h = 0;
            guard_lock_v = 1;
            break;

          case 3:  // left
            guard_lock_h = -1;
            guard_lock_v = 0;
            break;

          case 4:  // right
            guard_lock_h = 1;
            guard_lock_v = 0;
            break;
        }
        for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==0 && ice_tile_v
>=0;ice_tile_v--) {
          if((ice_tile_v == gtile_v + guard_lock_v && ice_tile_h== gtile_h +
guard_lock_h) || (ice_tile_v == gtile_v && ice_tile_h== gtile_h))
            break;
            else
             { mapping[gamelevel][ice_tile_v][ice_tile_h]=1;
```

```
            IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'
            IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'

            for(j=0;j<70000;j++);// add delay
          }
        }
        }

                    }
  else if (sprite_mode == 2){              // face left
    if (mapping[gamelevel][tile_v][tile_h-1] == 1){
       IOWR_WM8731_DATA(WM8731_BASE,31,2);
        ice_tile_v=tile_v;ice_tile_h=tile_h-1;
        for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==1 && ice_tile_h
>=0;ice_tile_h--)
          { mapping[gamelevel][ice_tile_v][ice_tile_h]=0;
            IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'
            IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'

            for(j=0;j<70000;j++);// add delay
          }
                          }
    else if (mapping[gamelevel][tile_v][tile_h-1] == 0){
       ice_tile_v=tile_v;
       ice_tile_h=tile_h-1;
```

```
IOWR_WM8731_DATA(WM8731_BASE,31,2);
switch(guard_lock_flag){
    case 0:
   guard_lock_h = 0;
   guard_lock_v = 0;
     break;

   case 1:  // up
     guard_lock_h = 0;
     guard_lock_v = -1;
     break;

   case 2:  // down
     guard_lock_h = 0;
     guard_lock_v = 1;
     break;

   case 3:  // left
     guard_lock_h = -1;
     guard_lock_v = 0;
     break;

   case 4:  // right
     guard_lock_h = 1;
```

```
            guard_lock_v = 0;

            break;

          }

         for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==0&& ice_tile_h
>=0;ice_tile_h--)   {

            if((ice_tile_v == gtile_v + guard_lock_v && ice_tile_h == gtile_h +
guard_lock_h) || (ice_tile_v == gtile_v && ice_tile_h== gtile_h))

            break;

            else

             { mapping[gamelevel][ice_tile_v][ice_tile_h]=1;

             IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'

             IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'



             for(j=0;j<70000;j++);// add delay

            }

          }

          }

                      }

   else {                              // face right

     if (mapping[gamelevel][tile_v][tile_h+1] == 1){

      IOWR_WM8731_DATA(WM8731_BASE,31,2);

          ice_tile_v=tile_v;ice_tile_h=tile_h+1;

          for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==1 && ice_tile_h
<20;ice_tile_h++)

            { mapping[gamelevel][ice_tile_v][ice_tile_h]=0;

             IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'
```

```
        IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'


        for(j=0;j<70000;j++);// add delay
       }
                          }
else if (mapping[gamelevel][tile_v][tile_h+1] == 0){
   ice_tile_v=tile_v;
   ice_tile_h=tile_h+1;
   IOWR_WM8731_DATA(WM8731_BASE,31,2);
   switch(guard_lock_flag){
      case 0:
      guard_lock_h = 0;
      guard_lock_v = 0;
        break;


      case 1:  // up
        guard_lock_h = 0;
        guard_lock_v = -1;
        break;


      case 2:  // down
        guard_lock_h = 0;
        guard_lock_v = 1;
        break;
```

```
            case 3:  // left
              guard_lock_h = -1;
              guard_lock_v = 0;
              break;

            case 4:  // right
              guard_lock_h = 1;
              guard_lock_v = 0;
              break;
          }
          for (;mapping[gamelevel][ice_tile_v][ice_tile_h]==0 && ice_tile_h
<20;ice_tile_h++)  {
            if((ice_tile_v == gtile_v + guard_lock_v && ice_tile_h== gtile_h +
guard_lock_h) || (ice_tile_v == gtile_v && ice_tile_h== gtile_h))
              break;
              else
               { mapping[gamelevel][ice_tile_v][ice_tile_h]=1;
               IOWR_VGA_DATA(VGA_BASE,4,ice_tile_v);//address'00100'
               IOWR_VGA_DATA(VGA_BASE,5,ice_tile_h);//address'00101'

               for(j=0;j<70000;j++);// add delay
              }
          }
          }
```

```
      }

 }
```

```
 int lv[41] =
{6,7,8,9,10,10,10,6,6,7,8,9,10,10,10,9,8,7,6,6,6,6,7,8,8,8,9,10,10,10,6,6,6,7,8,8,8,9,1
0,10,10};

 int lh[41] =
{3,3,3,3,3,4,5,8,7,7,7,7,7,8,9,9,9,9,9,13,12,11,11,11,12,13,13,13,12,11,17,16,15,15,1
7,16,15,15,15,16,17};



 int wv[35] =
{6,7,8,9,10,7,8,9,10,9,8,7,6,6,6,6,7,8,9,10,10,10,10,9,8,7,6,7,8,9,10,9,8,7,6};

 int wh[35] =
{3,3,3,3,4,5,5,5,6,7,7,7,7,9,10,11,10,10,10,9,10,11,13,13,13,13,13,14,15,16,17,17,17,
17,17};



 void gamewin(){

    int i,j;

    int counter;

    int m;

    gamemode = 2;    //win

    for (i=5;i<13;i++)

      for (j=0;j<20;j++)
```

```
   { if(mapping[2][i][j]== 1){ mapping[2][i][j]=0;
                        IOWR_VGA_DATA(VGA_BASE,4,i);
                        IOWR_VGA_DATA(VGA_BASE,5,j);
                       }
     else {}

   }
for(counter = 0; counter < 35; counter ++){
if(mapping[gamelevel][wv[counter]][wh[counter]]== 2){          //clear banana
 mapping[gamelevel][wv[counter]][wh[counter]] = 0;
IOWR_VGA_DATA(VGA_BASE,12,wv[counter]);//address'00100'
IOWR_VGA_DATA(VGA_BASE,13,wh[counter]);//address'00101'
}
if(mapping[gamelevel][wv[counter]][wh[counter]]== 3){          //clear grape
 mapping[gamelevel][wv[counter]][wh[counter]] = 0;
IOWR_VGA_DATA(VGA_BASE,6,wv[counter]);//address'00100'
IOWR_VGA_DATA(VGA_BASE,7,wh[counter]);//address'00101'
}
if(mapping[gamelevel][wv[counter]][wh[counter]]== 1){          //clear grape
 mapping[gamelevel][wv[counter]][wh[counter]] = 0;
  continue;
}

IOWR_VGA_DATA(VGA_BASE,4,wv[counter]);//address'00100'
```

```
    IOWR_VGA_DATA(VGA_BASE,5,wh[counter]);//address'00101'
//     for(m = 0 ; m < 10000; m ++){};


    }
 }


 void gameover(){
    int counter;
    int m;
    int i,j;
     for (i=5;i<13;i++)
      for (j=0;j<20;j++)
        { if(mapping[gamelevel][i][j]== 1){ mapping[gamelevel][i][j]=0;
                          IOWR_VGA_DATA(VGA_BASE,4,i);
                          IOWR_VGA_DATA(VGA_BASE,5,j);
                         }
          else {}


        }
    for(counter = 0; counter < 41; counter ++){
    if(mapping[gamelevel][lv[counter]][lh[counter]]== 2){          //clear banana
     mapping[gamelevel][lv[counter]][lh[counter]] = 0;
    IOWR_VGA_DATA(VGA_BASE,12,lv[counter]);//address'00100'
    IOWR_VGA_DATA(VGA_BASE,13,lh[counter]);//address'00101'
```

```
    }
    if(mapping[gamelevel][lv[counter]][lh[counter]]== 3){          //clear grape
     mapping[gamelevel][lv[counter]][lh[counter]] = 0;
    IOWR_VGA_DATA(VGA_BASE,6,lv[counter]);//address'00100'
    IOWR_VGA_DATA(VGA_BASE,7,lh[counter]);//address'00101'
    }

     if(mapping[gamelevel][wv[counter]][wh[counter]]== 1){          //clear grape
     mapping[gamelevel][wv[counter]][wh[counter]] = 0;
     continue;
    }
    IOWR_VGA_DATA(VGA_BASE,4,lv[counter]);//address'00100'
    IOWR_VGA_DATA(VGA_BASE,5,lh[counter]);//address'00101'
//     for(m = 0 ; m < 10000; m ++);
    }
 }


 void reset(){
   gx=448,gy=32,gflag= 0;  //reset guard
///   int gamelevel=0,gamemode=0;//0 start 1 dead  2 win
//   int grecord=0,brecord=0; //record reset
    x = 0;
    y = 32;
```

```
//    int sprite_mode = 0;
   IOWR_VGA_DATA(VGA_BASE,29,0);//address'00100'
 }


 int abs(int x){
   if(x < 0)
    return -x;
    else
    return x;
 }
```