PAC-XON

CSEE 4840 Embedded System Design

Dongwei Ge (dg2563 EE

Bo Liang (bl2369) ME

Jie Cai (jc3480) EE

Content

1. Introduction..2
2. Design...4
2.1 Game Logic..4
2.2 Software 4
2.3 Hardware..6
2.3.1 PS Input..6
2.3.2 Video Controller...6
2.3.3 Audio Controller...9
3. Conclusion 10
4. Codes... 11
4.1 Hardware 11
4.1.2 de2_sram_controller.vhd 11
4.1.3 de2_ps2.vhd…….. 12
4.1.4 de2_vga_raster.vhd... 18
4.1.5 tone_generateor.vhd.. 72
4.1.6 lab3_audio.vhd.. 79
4.2 Software... 89
4.2.1 hello_world.c….. 89

1. <u>Introduction</u>

Our project is to design a video game that consists of a combination of custom designed hardware and software, which will take everything we have learned so far in the past 2 months. In the hardware part, we will design a VGA controller, a CPU, a RAM controller and a interface between the hardware and the software.

To implement our project, VHDL and C programming language are both must for hardware part and software part respectively. With them we can handle inputs from the keyboard to the video display output on the screen.

We play to use the VGA output of the Altera board to present the game's graphics. Images will be initialized in hardware and ghosts' movement done in software. The software will also control game logic. Player will play the game with the "⇨ ⇦ ⇧⇩ " keys on the PS2 keyboard.

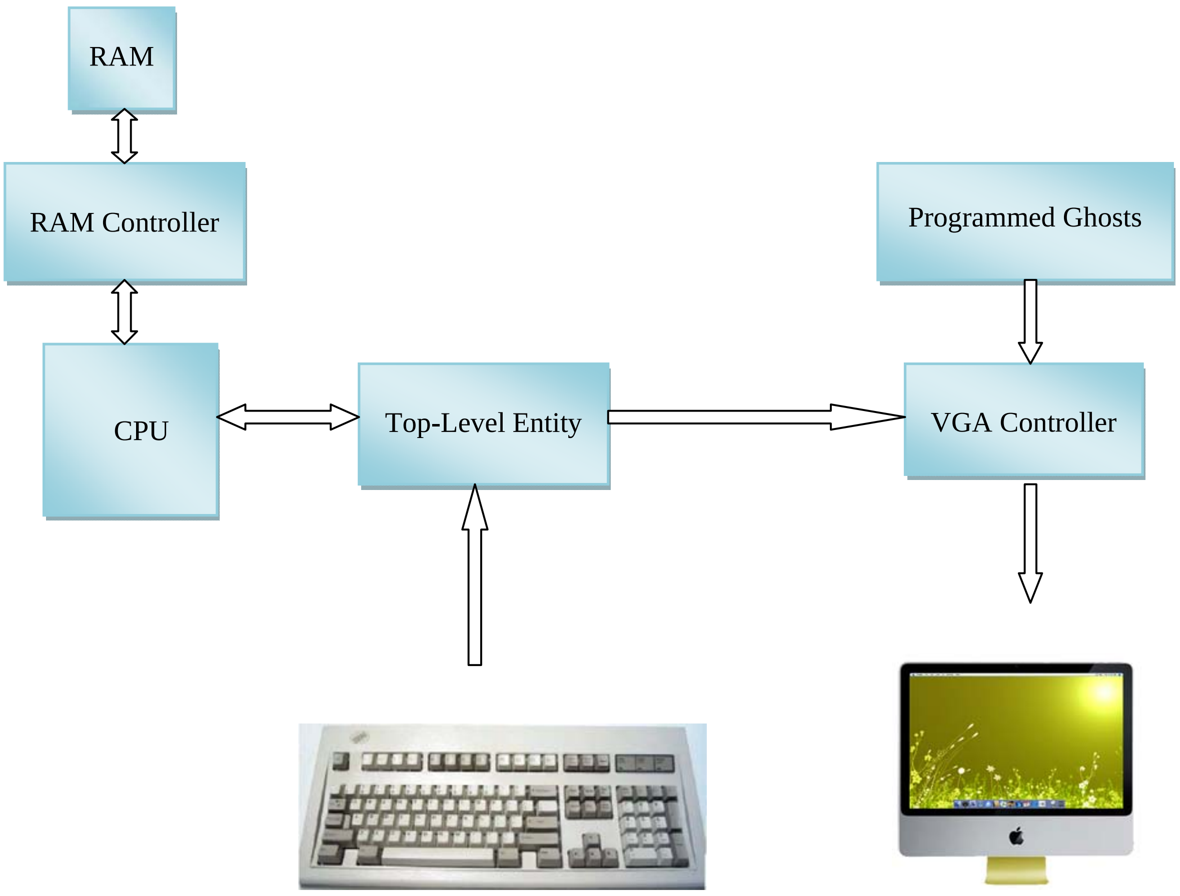


Fig.1 The block diagram of our project

2. Design

2.1 Game Logic

The game is similar to the original PAC-XON game: Each player controls a Pac-Man with four lives at the beginning of the game. When the game starts, Pac-Man can move freely from the top-left. Player must fill empty space and capture ghosts by building wall. As soon as the player fills 80% or more empty space he will go to the next level. Players must also beware of the two ghosts. If one touches the wall the pac man is building or catches the pac man, he will lose a life.

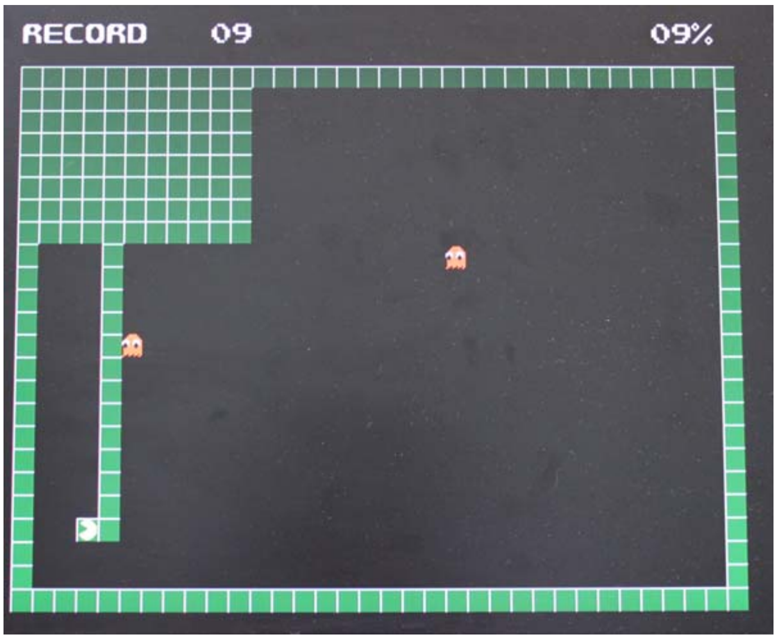


Fig.1 Our PAC-XON Game shot

2.2 Software

Here is the flow chart of our game. The movement of pac-man is designed tile by tile, the movement of ghosts are design pixel by pixel.

We can see how the states change in the game area from Fig. 2

🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	126	99	98	🟧								🟧	19	20	🟩
🟩	125	100	97	🟧								🟧	18	21	🟩
🟩	124	101	96	🟧								🟧	17	22	🟩
🟩	123	102	95	🟧		🟧	🟧	🟧	🟧	🟧		🟧	16	23	🟩
🟩	122	103	94	🟧		🟧	65	2	3	🟧		🟧	15	24	🟩
🟩	121	104	93	🟧		🟧	64	1	4	🟧		🟧	14	25	🟩
🟩	120	105	92	🟧		🟧	63	0	5	🟧		🟧	13	26	🟩
🟩	119	106	91	🟧		🟧	62	61	6	🟧	🟧	🟧	12	27	🟩
🟩	118	107	90	🟧		🟧	66	60	7	8	9	10	11	28	🟩
🟩	117	108	89	🟧		🟧	67	59	58	49	48	39	38	29	🟩
🟩	116	109	88	🟧	🟧	🟧	68	69	57	50	47	40	37	30	🟩
🟩	115	110	87	82	81	76	75	70	56	51	46	41	36	31	🟩
🟩	114	111	86	83	80	77	74	71	55	52	45	42	35	32	🟩
🟩	113	112	85	84	79	78	73	72	54	53	44	43	34	33	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

WALL
PATH
GHOST

Fig.2 States in our Game

1.find the gost

2.if UP==YES, go to (2); if NOT, go to (3)

3.if RIGHT==YES, go to (2); if NOT, go to (4)

4.if DOWN==YES, go to (2); if NOT, go to (5)

5.if LEFT==YES, go to (2); if NOT, go to (6)

6.if Now==GHOST, finish; if NOT, position--, go to (2)

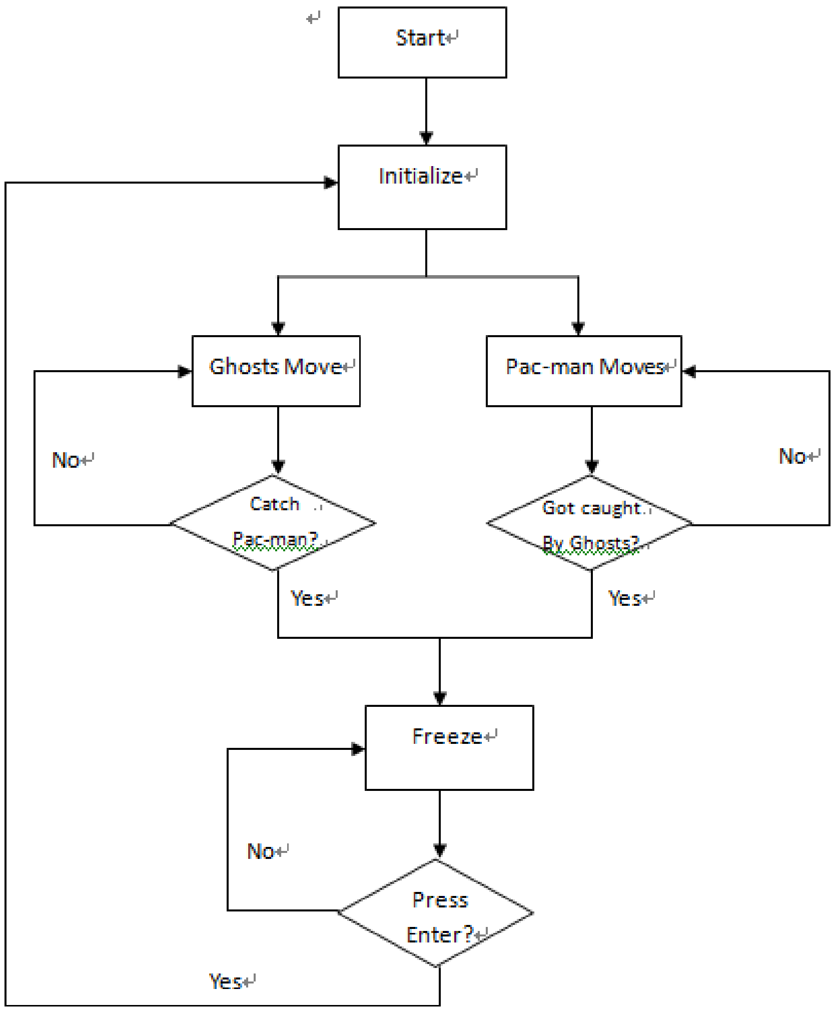


Fig. 3 Game Logic Flow Chart

2.3 Hardware

2.3.1 PS2 Input Controller

In our design, only four keys from the PS2 are used, "⇨ ⇦ ⇧ ⇩", which control the moving direction of the Pac-Man. The PS2 input controller is linked to the top level file of the project so that it can finally implement the movement of the Pac-Man. The movement of the Pac-Man related to the direction of the 4 keys are implemented by C programming language.

2.3.2 Video Controller

In our labs, we have learned to implement a live video display. We were able to store our character sets and graphics on the FPGA's SRAM. This allowed us to save valuable time

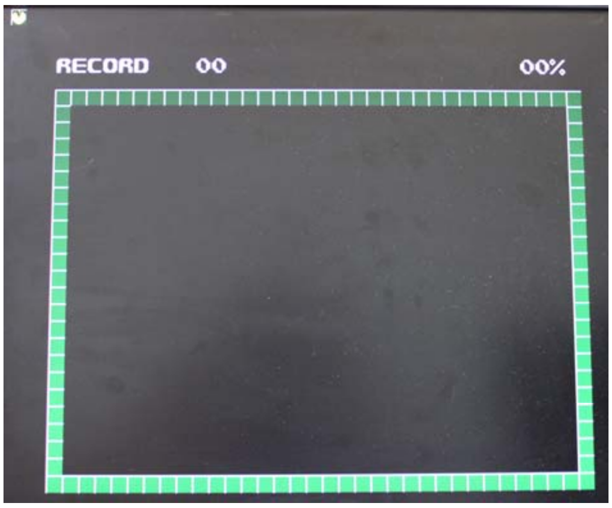
in drawing each character one by one by calling pre-existing sprite graphics.
Video (at VGA resolution) for PAC-XON posed a number of design challenges because of the number of concurrent tasks. There are 5 main components to video, which can be divided into 2 types of video. Tile graphics are displayed on 16-pixel boundaries on the screen, and are displayed in monochrome. On the other hand, sprite graphics are displayed at any arbitrary pixel location and can be poly-chromatic.

Component	Type	Requirement
Enclosure Wall	Sprite	Draw and Display by Hardware
Tiles	Sprite	Update by Software
Score	Tile/Sprite	Read from Software
Pac-Man	Sprite	Draw by Software & Display by Software
Ghosts	Sprite	Draw by Software & Display by Software

Fig.1 Our PAC-XON Game shot

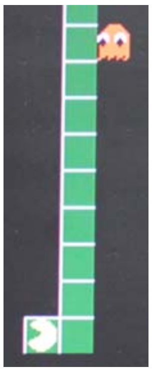
We used this part of our lab to display the enclosure wall and tail tiles drew by Pac-Man, the score and green wall square graphics on the screen.

Enclosure Wall
The Enclosure Wallis defined as an 32*42 RAM, which is always displays on the screen. Since each tile is 16x16 pixels, the Enclosure Wall definition covers almost the entire screen. Each bit in the RAM defines whether that tile on the screen is an unpassable wall ('1') or a navigable path ('0'). Each Enclosure tile is drew pixel by pixel ,allowing 4 bits per pixel to represent 1of the 7 colors we have chosen as out palette which allows us to multiple colors.



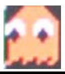
Tail Tile

The Tail Tile is defined as a 16*16 RAM, each unit is actually 1 pixel on the screen which has to be represented by one color, and it is drew in the same way the Enclosure Wall Tile. As the Pac-Man moves, the path it passed are changed from navigable path to Tail Tile, resulting the change of the 32*42 RAM, that is, the state of the unit in the RAM change from "0" to "1" . In reality, each move the Pac-Man made will refresh the 32*42 RAM for once.



Score

The Score are defined as a RAM of 3 bit, each bit is represented by a std_logic_vector of 4 bits (array (2 downto 0) of std_logic_vector (3 downto 0)). Each bit of the score can be displayed the digits from 0 to 9, so that it is possible to display the total scores of Pac-Man on screen. The process which checks tile locations checks for certain locations on screen. By reading values from memory, software can update the scores. Besides, we also add a previous high record, and it is the challenge for the players.

Ghost and Pac-Man 

Since a sprite can be at any arbitrary position, some special machinery is required to determine pixel locations. A process exists which checks the location of each sprite and if within the 16-pixel block formed by the coordinates, calculates the displacement of the currently scanned pixel to determine a particular 7-bit value representing the color of that pixel. These 3-bit values are then looked up by the pixel scanning process in a 7-color table. Each sprite uses its own color-table, enabling each sprite to use a variety of colors. Beside, as the pac-man changes its direction, we draw four sprites for the four direction pacman.

Pac-Man and Ghosts' locations are read from memory, and controlled by software. The sprites work exactly in the manner they are described.

Address	Discription
0, 1	Pan-Man's position
2-23	Play Area
24	Reserved
25, 26	Ghost1's position
27, 28	Ghost2's position
29, 30	Score
31	Pac-man Direction
32, 33	Record

Fig.4 Memory Addresses for the Video Controller

2.3.3 Audio Controller

The audio part is similar with the FM sound synthesizer of the lab3,The basic FM equation is

$$x(t) = \sin(\omega_c t + I \sin(\omega_m t))$$

Where $x(t)$ is the amplitude at time t. ω_c, is the carrier frequency(the fundamental tone we hear), ω_m is the modulating frequency, and I is the modulation depth. The timbre of the sound is largely determined by the ratio ω_c / ω_m , which is generally set to an integer ratio .The fundamental frequency of musical notes follow an exponential scale. The A above middle C is 440 Hz, and going up an octave doubles the frequency.

We make the music in the VHDL part, create four different music for the software to use in the video game. Just change the modulating frequency and the modulation depth to make the musical notes we wanted. However, how long the time the musical note will last? We use a counter to control the time the notes will last. For example, the clock is 50MH, we make the counter count to be 50M, the note will last one second.

3. Conclusion

We believe that the project was completed successfully. Although the AI is not very complicated, we have created a changeable background and implement the movement of the Pac-Man and ghosts and game works well with nice character graphics. We never imagined that creating PAX-XON would be so hard and time consuming.

At first we created the background and draw the sprite, but what we expected didn't show up until professor mentioned one most important issue—***"Initialization"***. Each variable RAM we used in the game has to be initialized then it could be display or updated by software.

In addition, after we have successfully built the play area, and started to focus on the software part, the most difficult part in the software design is that when we try to change in the state of the tiles as the pac-man or ghosts move, it is difficult to distinguish which part should be filled and which part should be left due to the movement of the ghosts. Here we use the concept of "connected" in graphics, which is starting from the two ghosts, check the four sides of the moving ghosts, then expand the checking area until it reach the real tiles that have no ghosts in there. By this method, we can change the states of the tiles in the play area.

Furthermore, when we decided to add some sound to the game in order to make it more vivid, we came across a problem, we have to revise the port map to add some ports and signals from the Audio Controller, which took us a long time to check the ports one by one and rearranged them.

Finally, we completed our design and implement the game logic. We can play it now!

Thanks to our professor Stephen Edwards, our TA Baolin Shao and Scott Schuff.

4. Codes:

4.1 Hardware

File Listings:

de2_sram_controller.vhd // The SRAM Controller
de2_ps2.vhd // The PS2 Controller
de2_vga_raster.vhd // The VGA Controller
tone_generator// The Audio Controller
lab3_audio.vhd // The Top Level Entity

4.1.1 de2_sram_controller.vhd // The SRAM Controller

```
library ieee;
use ieee.std_logic_1164.all
entity de2_sram_controller is

  port (
          --signal avs_s1_clk: in std_logic;
    signal avs_s1_chipselect : in std_logic;
    signal avs_s1_write, avs_s1_read : in std_logic;
    signal avs_s1_address  :  in std_logic_vector(17 downto 0);
    signal avs_s1_readdata : out std_logic_vector(15 downto 0);
    signal avs_s1_writedata : in std_logic_vector(15 downto 0);
    signal avs_s1_byteenable : in std_logic_vector(1 downto 0);

    signal SRAM_DQ   : inout std_logic_vector(15 downto 0);
    signal SRAM_ADDR : out std_logic_vector(17 downto 0);
    signal SRAM_UB_N, SRAM_LB_N : out std_logic;
    signal SRAM_WE_N, SRAM_CE_N : out std_logic;
    signal SRAM_OE_N          : out std_logic
    );

end de2_sram_controller;

architecture dp of de2_sram_controller is
begin

  SRAM_DQ <= avs_s1_writedata when avs_s1_write = '1' else (others => 'Z');
  avs_s1_readdata <= SRAM_DQ;
  SRAM_ADDR <= avs_s1_address;
  SRAM_UB_N <= not avs_s1_byteenable(1);
  SRAM_LB_N <= not avs_s1_byteenable(0);
  SRAM_WE_N <= not avs_s1_write;
  SRAM_CE_N <= not avs_s1_chipselect;
  SRAM_OE_N <= not avs_s1_read;

end dp;
```

4.1.2 de2_ps2.vhd // The PS2 Controller

```
-------------------------------------------------------------------
--
-- Simple (receive-only) PS/2 controller for the Altera Avalon bus
--
-- Presents a two-word interface:
--
-- Byte 0: LSB is a status bit: 1 = data received, 0 = no new data
-- Byte 4: least significant byte is received data,
--         reading it clears the input register
--
-- Make sure "Slave addressing" in the interfaces tab of SOPC Builder's
-- "New Component" dialog is set to "Register" mode.
--
--
-- Stephen A. Edwards and Yingjian Gu
-- Columbia University, sedwards@cs.columbia.edu
--
-- From an original by Bert Cuzeau
-- (c) ALSE. http://www.alse-fr.com
--
-------------------------------------------------------------------


-- ---------------------------------------------
--   Simplified PS/2 Controller  (kbd, mouse...)
-- ---------------------------------------------
-- Only the Receive function is implemented !
-- (c) ALSE. http://www.alse-fr.com
-- Author : Bert Cuzeau.
-- Fully synchronous solution, same Filter on PS2_Clk.
-- Still as compact as "Plain_wrong"...
-- Possible improvement : add TIMEOUT on PS2_Clk while shifting
-- Note: PS2_Data is resynchronized though this should not be
-- necessary (qualified by Fall_Clk and does not change at that time).
-- Note the tricks to correctly interpret 'H' as '1' in RTL simulation.

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

entity PS2_Ctrl is
  port(
    Clk       : in  std_logic;  -- System Clock
    Reset     : in  std_logic;  -- System Reset
    PS2_Clk   : in  std_logic;  -- Keyboard Clock Line
    PS2_Data  : in  std_logic;  -- Keyboard Data Line
    DoRead    : in  std_logic;  -- From outside when reading the scan code
    Scan_Err  : out std_logic;  -- To outside : Parity or Overflow error
    Scan_DAV  : out std_logic;  -- To outside when a scan code has arrived
    Scan_Code : out unsigned(7 downto 0) -- Eight bits Data Out
    );
end PS2_Ctrl;

architecture rtl of PS2_Ctrl is

  signal PS2_Datr  : std_logic;

  subtype Filter_t is unsigned(7 downto 0);
  signal Filter    : Filter_t;
  signal Fall_Clk  : std_logic;
  signal Bit_Cnt   : unsigned (3 downto 0);
  signal Parity    : std_logic;
  signal Scan_DAVi : std_logic;

  signal S_Reg     : unsigned(8 downto 0);

  signal PS2_Clk_f : std_logic;

  Type   State_t is (Idle, Shifting);
  signal State : State_t;
```

```
begin

  Scan_DAV <= Scan_DAVi;

-- This filters digitally the raw clock signal coming from the keyboard :
--  * Eight consecutive PS2_Clk=1 makes the filtered_clock go high
--  * Eight consecutive PS2_Clk=0 makes the filtered_clock go low
-- Implies a (FilterSize+1) x Tsys_clock delay on Fall_Clk wrt Data
-- Also in charge of the re-synchronization of PS2_Data

  process (Clk)
  begin
    if rising_edge(Clk) then
      if Reset = '1' then
        PS2_Datr  <= '0';
        PS2_Clk_f <= '0';
        Filter    <= (others => '0');
        Fall_Clk  <= '0';
      else
        PS2_Datr <= PS2_Data and PS2_Data; -- also turns 'H' into '1'
        Fall_Clk <= '0';
        Filter   <= (PS2_Clk and PS2_CLK) & Filter(Filter'high downto 1);
        if Filter = Filter_t'(others=>'1') then
          PS2_Clk_f <= '1';
        elsif Filter = Filter_t'(others=>'0') then
          PS2_Clk_f <= '0';
          if PS2_Clk_f = '1' then
            Fall_Clk <= '1';
          end if;
        end if;
      end if;
    end if;
  end process;

-- This simple State Machine reads in the Serial Data
-- coming from the PS/2 peripheral.

  process(Clk)
```

```
begin
  if rising_edge(Clk) then
    if Reset = '1' then
      State     <= Idle;
      Bit_Cnt   <= (others => '0');
      S_Reg     <= (others => '0');
      Scan_Code <= (others => '0');
      Parity    <= '0';
      Scan_DAVi <= '0';
      Scan_Err  <= '0';
    else

      if DoRead = '1' then
        Scan_DAVi <= '0'; -- note: this assgnmnt can be overriden
      end if;

      case State is

        when Idle =>
          Parity  <= '0';
          Bit_Cnt <= (others => '0');
          -- note that we do not need to clear the Shift Register
          if Fall_Clk='1' and PS2_Datr='0' then -- Start bit
            Scan_Err <= '0';
            State <= Shifting;
          end if;

        when Shifting =>
          if Bit_Cnt >= 9 then
            if Fall_Clk = '1' then -- Stop Bit
              -- Error is (wrong Parity) or (Stop='0') or Overflow
              Scan_Err  <= (not Parity) or (not PS2_Datr) or Scan_DAVi;
              Scan_Davi <= '1';
              Scan_Code <= S_Reg(7 downto 0);
              State <= Idle;
            end if;
          elsif Fall_Clk = '1' then
            Bit_Cnt <= Bit_Cnt + 1;
            S_Reg <= PS2_Datr & S_Reg (S_Reg'high downto 1); -- Shift right
            Parity <= Parity xor PS2_Datr;
          end if;

        when others => -- never reached
          State <= Idle;

      end case;

      --Scan_Err <= '0'; -- to create a deliberate error

    end if;

  end if;

end process;

end rtl;

-----------------------------------------------------------------------------

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

entity de2_ps2 is

port (
  avs_s1_clk        : in std_logic;
  avs_s1_reset      : in std_logic;
  avs_s1_address    : in unsigned(0 downto 0);
  avs_s1_read       : in std_logic;
  avs_s1_chipselect : in std_logic;
  avs_s1_readdata   : out unsigned(7 downto 0);

  PS2_Clk           : in std_logic;
  PS2_Data          : in std_logic
  );
end de2_ps2;
```

```
architecture rtl of de2_ps2 is

  signal Data          : unsigned(7 downto 0);
  signal DataAvailable : std_logic;
  signal DoRead        : std_logic;
begin

  U1: entity work.PS2_CTRL port map(
    Clk       => avs_s1_clk,
    Reset     => avs_s1_reset,
    DoRead    => DoRead,
    PS2_Clk   => PS2_Clk,
    PS2_Data  => PS2_Data,
    Scan_Code => Data,
    Scan_DAV  => DataAvailable );

  process (avs_s1_clk)
  begin
    if rising_edge(avs_s1_clk) then
      DoRead <= avs_s1_read and avs_s1_chipselect and avs_s1_address(0);
    end if;
  end process;

  process (Data, DataAvailable, avs_s1_address, avs_s1_chipselect)
  begin
    if avs_s1_chipselect = '1' then
      if avs_s1_address(0) = '1' then
        avs_s1_readdata <= Data;
      else
        avs_s1_readdata <= "0000000" & DataAvailable;
      end if;
    else
      avs_s1_readdata <= "00000000";
    end if;
  end process;
end rtl;
```

4.1.3 de2_vga_raster.vhd // The VGA Controller

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity de2_vga_raster is
  port (
    avs_s1_clk        : in  std_logic;
    avs_s1_reset_n    : in  std_logic;
    avs_s1_read       : in  std_logic;
    avs_s1_write      : in  std_logic;
    avs_s1_chipselect : in  std_logic;
    avs_s1_address    : in  std_logic_vector(6 downto 0);
    avs_s1_readdata   : out std_logic_vector(31 downto 0);
    avs_s1_writedata  : in  std_logic_vector(31 downto 0);

        reset_n         : in std_logic;                              -- More reset!
        CLK_25          : in std_logic;
    VGA_CLK   : out std_logic;                  -- Clock
    VGA_HS    : out std_logic;                  -- H_SYNC
    VGA_VS    : out std_logic;                  -- V_SYNC
    VGA_BLANK : out std_logic;                  -- BLANK
    VGA_SYNC  : out std_logic;                  -- SYNC
    VGA_R     : out std_logic_vector(9 downto 0); -- Red[9:0]
    VGA_G     : out std_logic_vector(9 downto 0); -- Green[9:0]
    VGA_B     : out std_logic_vector(9 downto 0)  -- Blue[9:0]
  );
end de2_vga_raster;

architecture rtl of  de2_vga_raster is

signal clk : std_logic:='0';

--Declare our various RAMs and ROMs
-----------------------------------------------------------------------
-----------------------------------------------------------------------
type wall_ram_type is array(0 to 31) of std_logic_vector(41 downto 0);
-----------------------------------------------------------------------
```

```
------------------------------------------------------------------------
type color_ram_type is array (6 downto 0) of std_logic_vector(29 downto 0);
type sprite_ram_line_type is array(0 to 15) of std_logic_vector(0 to 15);
type pacman_sprite_ram_line_type is array(0 to 15) of std_logic_vector(0 to 2);
type pacman_sprite_ram_type is array(0 to 15) of pacman_sprite_ram_line_type;
type num_sprite_ram_type is array(0 to 11) of sprite_ram_line_type;


signal num_sprite_ram: num_sprite_ram_type :=
(("0000000000000000","0000011111100000","0000011111100000","000110000111100
0","0001100001111000","0111100000011110","0111100000011110","01111000000111
10","0111100000011110","0111100000011110","0111100000011110","0001111000011
000","0001111000011000","0000011111100000","0000011111100000","000000000000
0000"),

("0000000000000000","0000001111000000","0000001111000000","0000111111000000
","0000111111000000","0000001111000000","0000001111000000","000000111100000
0","0000001111000000","0000001111000000","0000001111000000","00000011110000
00","0000001111000000","0011111111111100","0011111111111100","0000000000000
000"),

("0000000000000000","0001111111111000","0001111111111000","0111100000011110
","0111100000011110","0000000001111110","0000000001111110","000001111111100
0","0000011111111000","0001111111100000","0001111111100000","01111110000000
00","0111111000000000","0111111111111110","0111111111111110","0000000000000
000"),

("0000000000000000","0000011111111000","0000011111111000","0001100001111000
","0001100001111000","0000000001111000","0000000001111000","000000011110000
0","0000000111100000","0000000001111110","0000000001111110","01100000000111
10","0110000000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0000000111111000","0000000111111000","0000011111111000
","0000011111111000","0001111001111000","0001111001111000","011110000111100
0","0111100001111000","0111111111111110","0000000001111000","00000000011110
00","0000000001111000","0000000001111000","0000000001111000","0000000000000
000"),

("0000000000000000","0111111111111000","0111111111111000","0111100000000000
","0111100000000000","0111111111111000","0111111111111000","000000000001111
0","0000000000011110","0000000000011110","0000000000011110","01111000000111
10","0111100000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0001111111111000","0001111111111000","0111100000000000
","0111100000000000","0111111111111000","0111111111111000","011110000001111
0","0111100000011110","0111100000011110","0111100000011110","01111000000111
10","0111100000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0111111111111110","0111111111111110","0111100000011110
","0111100000011110","0000000001111000","0000000001111000","000000011110000
0","0000000111100000","0000011110000000","0000011110000000","00000111100000
00","0000011110000000","0000011110000000","0000011110000000","0000000000000
000"),

("0000000000000000","0001111111100000","0001111111100000","0111100000011000
","0111100000011000","0111111000011000","0111111000011000","000111111110000
0","0001111111100000","0110000111111110","0110000111111110","01100000000111
10","0110000000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0001111111111000","0001111111111000","0111100000011110
","0111100000011110","0111100000011110","0111100000011110","000111111111111
0","0001111111111110","0000000000011110","0000000000011110","00000000011110
00","0000000001111000","0001111111100000","0001111111100000","0000000000000
000"),

("0000000000000000","0000000000000000","0000000000000000","0000000000000000
","0000000000000000","0000000000000000","0000000000000000","000000000000000
0","0000000000000000","0000000000000000","0000000000000000","00000000000000
00","0000000000000000","0000000000000000","0000000000000000","0000000000000
000"),

("0000000000000000","0011000000000010","0100100000000100","0100100000001000
","0011000000010000","0000000000100000","0000000001000000","000000001000000
0","0000000100000000","0000001000000000","0000010000000000","00001000000011
00","0001000000010010","0010000000010010","0100000000001100","0000000000000
000"));
```

```
signal char_sprite_ram: num_sprite_ram_type :=
(("0000000000000000","0000011111100000","0000011111100000","000110000111100
0","0001100001111000","0111100000011110","0111100000011110","01111000000111
10","0111100000011110","0111100000011110","0111100000011110","0001111000011
000","0001111000011000","0000011111100000","0000011111100000","000000000000
0000"),

("0000000000000000","0000001111000000","0000001111000000","0000111111000000
","0000111111000000","0000001111000000","0000001111000000","000000111100000
0","0000001111000000","0000001111000000","0000001111000000","00000011110000
00","0000001111000000","0011111111111100","0011111111111100","0000000000000
000"),

("0000000000000000","0001111111111000","0001111111111000","0111100000011110
","0111100000011110","0000000001111110","0000000001111110","000001111111100
0","0000011111111000","0001111111100000","0001111111100000","01111110000000
00","0111111000000000","0111111111111110","0111111111111110","0000000000000
000"),

("0000000000000000","0000011111111000","0000011111111000","0001100001111000
","0001100001111000","0000000001111000","0000000001111000","000000011110000
0","0000000111100000","0000000001111110","0000000001111110","01100000000111
10","0110000000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0000000111111000","0000000111111000","0000011111111000
","0000011111111000","0001111001111000","0001111001111000","011110000111100
0","0111100001111000","0111111111111110","0000000001111000","00000000011110
00","0000000001111000","0000000001111000","0000000001111000","0000000000000
000"),

("0000000000000000","0111111111111000","0111111111111000","0111100000000000
","0111100000000000","0111111111111000","0111111111111000","000000000001111
0","0000000000011110","0000000000011110","0000000000011110","01111000000111
10","0111100000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0001111111111000","0001111111111000","0111100000000000
","0111100000000000","0111111111111000","0111111111111000","011110000001111
0","0111100000011110","0111100000011110","0111100000011110","01111000000111
10","0111100000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0111111111111110","0111111111111110","0111100000011110
","0111100000011110","0000000001111000","0000000001111000","000000011110000
0","0000000111100000","0000011110000000","0000011110000000","00000111100000
00","0000011110000000","0000011110000000","0000011110000000","0000000000000
000"),

("0000000000000000","0001111111100000","0001111111100000","0111100000011000
","0111100000011000","0111111000011000","0111111000011000","000111111110000
0","0001111111100000","0110000111111110","0110000111111110","01100000000111
10","0110000000011110","0001111111111000","0001111111111000","0000000000000
000"),

("0000000000000000","0001111111111000","0001111111111000","0111100000011110
","0111100000011110","0111100000011110","0111100000011110","000111111111111
0","0001111111111110","0000000000011110","0000000000011110","00000000011110
00","0000000001111000","0001111111100000","0001111111100000","0000000000000
000"),

("0000000000000000","0000000000000000","0000000000000000","0000000000000000
","0000000000000000","0000000000000000","0000000000000000","000000000000000
0","0000000000000000","0000000000000000","0000000000000000","00000000000000
00","0000000000000000","0000000000000000","0000000000000000","0000000000000
000"),

("0000000000000000","0011000000000010","0100100000000100","0100100000001000
","0011000000010000","0000000000100000","0000000001000000","000000001000000
0","0000000100000000","0000001000000000","0000010000000000","00001000000011
00","0001000000010010","0010000000010010","0100000000001100","0000000000000
000"));

signal score_ram_1, score_ram_2, score_ram_temp_1, score_ram_temp_2,
record_ram_1,
record_ram_2, record_ram_temp_1, record_ram_temp_2: std_logic_vector(3 downto
0) :=
"0000";
```

```
signal RAM : wall_ram_type := ("0000000000000000000000000000000000000000",
                                "0000000000000000000000000000000000000000",
                                "0000000000000000000000000000000000000000",
                                "0000000000000000000000000000000000000000",
                                "0000000000000000000000000000000000000000",
                               "0000011111111111111111111111111111111000",
                               "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                                "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                                "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                              "0000010000000000000000000000000000001000",
                                "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                              "0000010000000000000000000000000000001000",
                                "0000010000000000000000000000000000001000",
                              "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                                "0000010000000000000000000000000000001000",
                             "0000010000000000000000000000000000001000",
                                "0000010000000000000000000000000000001000",
                                "0000010000000000000000000000000000001000",
                                  "0000010000000000000000000000000000001000",
                                 "0000010000000000000000000000000000001000",
                               "0000010000000000000000000000000000001000",
                               "0000011111111111111111111111111111111000",
                               "0000000000000000000000000000000000000000",
                               "0000000000000000000000000000000000000000",
                              "0000000000000000000000000000000000000000");
----------------------------------------------------------
----------------------------------------------------------
signal RAM_temp : wall_ram_type :=
("0000000000000000000000000000000000000000",
                               "0000000000000000000000000000000000000000",
                               "0000000000000000000000000000000000000000",
                               "0000000000000000000000000000000000000000",
                                "0000000000000000000000000000000000000000",
                                        "0000011111111111111111111111111111111000",
                                     "0000010000000000000000000000000000001000",
                                      "0000010000000000000000000000000000001000",
                                         "0000010000000000000000000000000000001000",
                                         "0000010000000000000000000000000000001000",
                                        "0000010000000000000000000000000000001000",
                                        "0000010000000000000000000000000000001000",
                                         "0000010000000000000000000000000000001000",
                                        "0000010000000000000000000000000000001000",
                                         "0000010000000000000000000000000000001000",
                                        "0000010000000000000000000000000000001000",
                                        "0000010000000000000000000000000000001000",
                                     "0000010000000000000000000000000000001000",
                                     "0000010000000000000000000000000000001000",
                                      "0000010000000000000000000000000000001000",
                                       "0000010000000000000000000000000000001000",
                                       "0000010000000000000000000000000000001000",
                                         "0000010000000000000000000000000000001000",
                                      "0000010000000000000000000000000000001000",
                                   "0000010000000000000000000000000000001000",
                                         "0000010000000000000000000000000000001000",
                                       "0000010000000000000000000000000000001000",
                                        "0000010000000000000000000000000000001000",
                                        "0000011111111111111111111111111111111000",
                                        "0000000000000000000000000000000000000000",
                                         "0000000000000000000000000000000000000000",
                                       "0000000000000000000000000000000000000000");
--------------------------------------------------------
--------------------------------------------------------
signal color_ram: color_ram_type;

signal square_sprite_ram: pacman_sprite_ram_type;
signal black_sprite_ram: pacman_sprite_ram_type;
signal pacman_sprite_0_ram: pacman_sprite_ram_type;
signal pacman_sprite_up_ram: pacman_sprite_ram_type;
signal pacman_sprite_down_ram: pacman_sprite_ram_type;
signal pacman_sprite_left_ram: pacman_sprite_ram_type;
signal pacman_sprite_right_ram: pacman_sprite_ram_type;
signal ghost_sprite_1_ram: pacman_sprite_ram_type;
signal ghost_sprite_2_ram: pacman_sprite_ram_type;
signal ram_address, display_address : std_logic_vector(6 downto 0):="0000000";
signal show_square: std_logic:='0';

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;

-- Signals for the video controller
signal Hcount : std_logic_vector(9 downto 0);  -- Horizontal position (0-800)
signal Vcount : std_logic_vector(9 downto 0);  -- Vertical position (0-524)
signal EndOfLine, EndOfField : std_logic;

signal vga_hblank, vga_hsync,
 vga_vblank, vga_vsync : std_logic;  -- Sync. signals

signal visible_xpos: std_logic_vector(9 downto 0):="0000000000";
signal visible_ypos: std_logic_vector(9 downto 0):="0000000000";

signal square_sprite_x: std_logic_vector(3 downto 0):="0000";
signal square_sprite_y: std_logic_vector(3 downto 0):="0000";

signal pacman_x: std_logic_vector(9 downto 0):="0000011001";
signal pacman_y: std_logic_vector(9 downto 0):="0000100101";
signal ghost_1_x: std_logic_vector(9 downto 0):="0000110010";
signal ghost_1_y: std_logic_vector(9 downto 0):="0000111001";
signal ghost_2_x: std_logic_vector(9 downto 0):="0000111001";
signal ghost_2_y: std_logic_vector(9 downto 0):="0000111101";
signal pacman_direction : std_logic_vector(2 downto 0):="000";
signal square_x: std_logic_vector(5 downto 0):="000000";
signal square_y: std_logic_vector(5 downto 0):="000000";
signal show_wall_square : std_logic:='0';
signal print_char : std_logic:='0';
signal draw_pacman_0, draw_ghost_1, draw_ghost_2 : std_logic:='0';
signal wall_square_sprite_from_RAM : pacman_sprite_ram_type;
signal wall_square_state_from_RAM : std_logic:='0';

--------------------------SCORE CHAR INITIALIZATION : 0----------------------
----------------------------------------------------------------------------
signal current_char_sprite_from_RAM : sprite_ram_line_type :=
("0000000000000000","0000001111000000","0000001111000000","0000111111000000
","0000111111000000","0000001111000000","0000001111000000","000000111100000
0","0000001111000000","0000001111000000","0000001111000000","00000011110000
00","0000001111000000","0011111111111100","0011111111111100","0000000000000
000");

  signal rhs : std_logic_vector(9 downto 0):="0000000000";
  signal rvs : std_logic_vector(9 downto 0):="0000000000";
  signal ghost1_rhs : std_logic_vector(9 downto 0):="0000000000";
  signal ghost1_rvs : std_logic_vector(9 downto 0):="0000000000";
  signal ghost2_rhs : std_logic_vector(9 downto 0):="0000000000";
  signal ghost2_rvs : std_logic_vector(9 downto 0):="0000000000";
  signal pacman_direction_temp : std_logic_vector(2 downto 0):="000";
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

begin

  clk <= avs_s1_clk;
  ram_address <= avs_s1_address;
  visible_xpos <= Hcount - HSYNC - HBACK_PORCH;
  square_x <= visible_xpos(9 downto 4);
  square_sprite_x <= visible_xpos(3 downto 0);



  DrawWall: process(clk)
  begin
    if wall_square_state_from_RAM = '1' then
       show_wall_square <= '1';

        else
```

```
            show_wall_square <= '0';
    end if;
end process;

  Drawscore: process(clk)
  begin
    if square_x = "100001" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= num_sprite_ram(conv_integer(score_ram_2));
           --current_char_sprite_from_RAM <= num_sprite_ram(8);
    elsif square_x = "100010" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= num_sprite_ram(conv_integer(score_ram_1));
           --current_char_sprite_from_RAM <= num_sprite_ram(0);
    elsif square_x = "100011" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= num_sprite_ram(11);
        elsif square_x = "000011" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= char_sprite_ram(0);
        elsif square_x = "000100" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= char_sprite_ram(1);
        elsif square_x = "000101" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= char_sprite_ram(2);
        elsif square_x = "000110" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= char_sprite_ram(3);
        elsif square_x = "000111" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= char_sprite_ram(0);
        elsif square_x = "001000" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= char_sprite_ram(4);
        elsif square_x = "001100" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= num_sprite_ram(conv_integer(record_ram_2));
        elsif square_x = "001101" and square_y = "000011" then
       print_char <= '1';
       current_char_sprite_from_RAM <= num_sprite_ram(conv_integer(record_ram_1));
    else
          print_char <= '0';
        end if;
end process;

 CheckDrawPacman: process (clk)
  begin
        if visible_xpos < pacman_x+16 and visible_xpos >= pacman_x and visible_ypos <
pacman_y+16 and visible_ypos >= pacman_y then
                draw_pacman_0 <= '1';
        else
                draw_pacman_0 <= '0';
        end if;
  end process;
CheckDrawGhost_1: process (clk)
 begin
 if visible_xpos < ghost_1_x+16 and visible_xpos >= ghost_1_x and visible_ypos <
ghost_1_y+16 and visible_ypos >= ghost_1_y then
                draw_ghost_1 <= '1';

        else
                draw_ghost_1 <= '0';
        end if;
  end process;

CheckDrawGhost_2: process (clk)
 begin
 if visible_xpos < ghost_2_x+16 and visible_xpos >= ghost_2_x and visible_ypos <
ghost_2_y+16 and visible_ypos >= ghost_2_y then
                draw_ghost_2 <= '1';

        else
                draw_ghost_2 <= '0';
        end if;
  end process;

  process (clk)
  begin
        --all of these registers end up being reduced, so no worries :-)  I'll try to clean it
```

up later, though.

```
  if clk'event and clk = '1' then

      num_sprite_ram(0)(0) <= "0000000000000000";
      num_sprite_ram(0)(1) <= "0000011111100000";
      num_sprite_ram(0)(2) <= "0000011111100000";
      num_sprite_ram(0)(3) <= "0001100001111000";
      num_sprite_ram(0)(4) <= "0001100001111000";
      num_sprite_ram(0)(5) <= "0111100000011110";
      num_sprite_ram(0)(6) <= "0111100000011110";
      num_sprite_ram(0)(7) <= "0111100000011110";
      num_sprite_ram(0)(8) <= "0111100000011110";
      num_sprite_ram(0)(9) <= "0111100000011110";
      num_sprite_ram(0)(10) <="0111100000011110";
      num_sprite_ram(0)(11) <="0001111000011000";
      num_sprite_ram(0)(12) <="0001111000011000";
      num_sprite_ram(0)(13) <="0000011111100000";
      num_sprite_ram(0)(14) <="0000011111100000";
      num_sprite_ram(0)(15) <="0000000000000000";


      num_sprite_ram(1)(0) <= "0000000000000000";
      num_sprite_ram(1)(1) <= "0000001111000000";
      num_sprite_ram(1)(2) <= "0000001111000000";
      num_sprite_ram(1)(3) <= "0000111111000000";
      num_sprite_ram(1)(4) <= "0000111111000000";
      num_sprite_ram(1)(5) <= "0000001111000000";
      num_sprite_ram(1)(6) <= "0000001111000000";
      num_sprite_ram(1)(7) <= "0000001111000000";
      num_sprite_ram(1)(8) <= "0000001111000000";
      num_sprite_ram(1)(9) <= "0000001111000000";
      num_sprite_ram(1)(10) <="0000001111000000";
      num_sprite_ram(1)(11) <="0000001111000000";
      num_sprite_ram(1)(12) <="0000001111000000";
      num_sprite_ram(1)(13) <="0011111111111100";
      num_sprite_ram(1)(14) <="0011111111111100";
      num_sprite_ram(1)(15) <="0000000000000000";

      num_sprite_ram(2)(0) <= "0000000000000000";
      num_sprite_ram(2)(1) <= "0001111111111000";
```

```
num_sprite_ram(2)(2) <= "0001111111111000";
num_sprite_ram(2)(3) <= "0111100000011110";
num_sprite_ram(2)(4) <= "0111100000011110";
num_sprite_ram(2)(5) <= "0000000001111110";
num_sprite_ram(2)(6) <= "0000000001111110";
num_sprite_ram(2)(7) <= "0000011111111000";
num_sprite_ram(2)(8) <= "0000011111111000";
num_sprite_ram(2)(9) <= "0001111111100000";
num_sprite_ram(2)(10) <="0001111111100000";
num_sprite_ram(2)(11) <="0111111000000000";
num_sprite_ram(2)(12) <="0111111000000000";
num_sprite_ram(2)(13) <="0111111111111110";
num_sprite_ram(2)(14) <="0111111111111110";
num_sprite_ram(2)(15) <="0000000000000000";

num_sprite_ram(3)(0) <= "0000000000000000";
num_sprite_ram(3)(1) <= "0000011111111000";
num_sprite_ram(3)(2) <= "0000011111111000";
num_sprite_ram(3)(3) <= "0001100001111000";
num_sprite_ram(3)(4) <= "0001100001111000";
num_sprite_ram(3)(5) <= "0000000001111000";
num_sprite_ram(3)(6) <= "0000000001111000";
num_sprite_ram(3)(7) <= "0000000111100000";
num_sprite_ram(3)(8) <= "0000000111100000";
num_sprite_ram(3)(9) <= "0000000001111110";
num_sprite_ram(3)(10) <="0000000001111110";
num_sprite_ram(3)(11) <="0110000000011110";
num_sprite_ram(3)(12) <="0110000000011110";
num_sprite_ram(3)(13) <="0001111111111000";
num_sprite_ram(3)(14) <="0001111111111000";
num_sprite_ram(3)(15) <="0000000000000000";

num_sprite_ram(4)(0) <= "0000000000000000";
num_sprite_ram(4)(1) <= "0000000111111000";
num_sprite_ram(4)(2) <= "0000000111111000";
num_sprite_ram(4)(3) <= "0000011111111000";
num_sprite_ram(4)(4) <= "0000011111111000";
num_sprite_ram(4)(5) <= "0001111001111000";
num_sprite_ram(4)(6) <= "0001111001111000";
num_sprite_ram(4)(7) <= "0111100001111000";
num_sprite_ram(4)(8) <= "0111100001111000";
num_sprite_ram(4)(9) <= "0111111111111110";
num_sprite_ram(4)(10) <="0000000001111000";
num_sprite_ram(4)(11) <="0000000001111000";
num_sprite_ram(4)(12) <="0000000001111000";
num_sprite_ram(4)(13) <="0000000001111000";
num_sprite_ram(4)(14) <="0000000001111000";
num_sprite_ram(4)(15) <="0000000000000000";

num_sprite_ram(5)(0) <= "0000000000000000";
num_sprite_ram(5)(1) <= "0111111111111000";
num_sprite_ram(5)(2) <= "0111111111111000";
num_sprite_ram(5)(3) <= "0111100000000000";
num_sprite_ram(5)(4) <= "0111100000000000";
num_sprite_ram(5)(5) <= "0111111111111000";
num_sprite_ram(5)(6) <= "0111111111111000";
num_sprite_ram(5)(7) <= "0000000000011110";
num_sprite_ram(5)(8) <= "0000000000011110";
num_sprite_ram(5)(9) <= "0000000000011110";
num_sprite_ram(5)(10) <="0000000000011110";
num_sprite_ram(5)(11) <="0111100000011110";
num_sprite_ram(5)(12) <="0111100000011110";
num_sprite_ram(5)(13) <="0001111111111000";
num_sprite_ram(5)(14) <="0001111111111000";
num_sprite_ram(5)(15) <="0000000000000000";

num_sprite_ram(6)(0) <= "0000000000000000";
num_sprite_ram(6)(1) <= "0001111111111000";
num_sprite_ram(6)(2) <= "0001111111111000";
num_sprite_ram(6)(3) <= "0111100000000000";
num_sprite_ram(6)(4) <= "0111100000000000";
num_sprite_ram(6)(5) <= "0111111111111000";
num_sprite_ram(6)(6) <= "0111111111111000";
num_sprite_ram(6)(7) <= "0111100000011110";
num_sprite_ram(6)(8) <= "0111100000011110";
num_sprite_ram(6)(9) <= "0111100000011110";
num_sprite_ram(6)(10) <="0111100000011110";
num_sprite_ram(6)(11) <="0111100000011110";
num_sprite_ram(6)(12) <="0111100000011110";
num_sprite_ram(6)(13) <="0001111111111000";
num_sprite_ram(6)(14) <="0001111111111000";
num_sprite_ram(6)(15) <="0000000000000000";

num_sprite_ram(7)(0) <= "0000000000000000";
num_sprite_ram(7)(1) <= "0111111111111110";
num_sprite_ram(7)(2) <= "0111111111111110";
num_sprite_ram(7)(3) <= "0111100000011110";
num_sprite_ram(7)(4) <= "0111100000011110";
num_sprite_ram(7)(5) <= "0000000001111000";
num_sprite_ram(7)(6) <= "0000000001111000";
num_sprite_ram(7)(7) <= "0000000111100000";
num_sprite_ram(7)(8) <= "0000000111100000";
num_sprite_ram(7)(9) <= "0000011110000000";
num_sprite_ram(7)(10) <="0000011110000000";
num_sprite_ram(7)(11) <="0000011110000000";
num_sprite_ram(7)(12) <="0000011110000000";
num_sprite_ram(7)(13) <="0000011110000000";
num_sprite_ram(7)(14) <="0000011110000000";
num_sprite_ram(7)(15) <="0000000000000000";

num_sprite_ram(8)(0) <= "0000000000000000";
num_sprite_ram(8)(1) <= "0001111111100000";
num_sprite_ram(8)(2) <= "0001111111100000";
num_sprite_ram(8)(3) <= "0111100000011000";
num_sprite_ram(8)(4) <= "0111100000011000";
num_sprite_ram(8)(5) <= "0111111000011000";
num_sprite_ram(8)(6) <= "0111111000011000";
num_sprite_ram(8)(7) <= "0001111111100000";
num_sprite_ram(8)(8) <= "0001111111100000";
num_sprite_ram(8)(9) <= "0110000111111110";
num_sprite_ram(8)(10) <="0110000111111110";
num_sprite_ram(8)(11) <="0110000000011110";
num_sprite_ram(8)(12) <="0110000000011110";
num_sprite_ram(8)(13) <="0001111111111000";
num_sprite_ram(8)(14) <="0001111111111000";
num_sprite_ram(8)(15) <="0000000000000000";

num_sprite_ram(9)(0) <= "0000000000000000";
num_sprite_ram(9)(1) <= "0001111111111000";
num_sprite_ram(9)(2) <= "0001111111111000";
num_sprite_ram(9)(3) <= "0111100000011110";
num_sprite_ram(9)(4) <= "0111100000011110";
num_sprite_ram(9)(5) <= "0111100000011110";
num_sprite_ram(9)(6) <= "0111100000011110";
num_sprite_ram(9)(7) <= "0001111111111110";
num_sprite_ram(9)(8) <= "0001111111111110";
num_sprite_ram(9)(9) <= "0000000000011110";
num_sprite_ram(9)(10) <="0000000000011110";
num_sprite_ram(9)(11) <="0000000001111000";
num_sprite_ram(9)(12) <="0000000001111000";
num_sprite_ram(9)(13) <="0001111111100000";
num_sprite_ram(9)(14) <="0001111111100000";
num_sprite_ram(9)(15) <="0000000000000000";

num_sprite_ram(10)(0) <= "0000000000000000";
num_sprite_ram(10)(1) <= "0000000000000000";
num_sprite_ram(10)(2) <= "0000000000000000";
num_sprite_ram(10)(3) <= "0000000000000000";
num_sprite_ram(10)(4) <= "0000000000000000";
num_sprite_ram(10)(5) <= "0000000000000000";
num_sprite_ram(10)(6) <= "0000000000000000";
num_sprite_ram(10)(7) <= "0000000000000000";
num_sprite_ram(10)(8) <= "0000000000000000";
num_sprite_ram(10)(9) <= "0000000000000000";
num_sprite_ram(10)(10) <="0000000000000000";
num_sprite_ram(10)(11) <="0000000000000000";
num_sprite_ram(10)(12) <="0000000000000000";
num_sprite_ram(10)(13) <="0000000000000000";
num_sprite_ram(10)(14) <="0000000000000000";
num_sprite_ram(10)(15) <="0000000000000000";

num_sprite_ram(11)(0) <= "0011000000000000";
num_sprite_ram(11)(1) <= "0111100000000111";
num_sprite_ram(11)(2) <= "1100110000001110";
num_sprite_ram(11)(3) <= "1100110000011100";
num_sprite_ram(11)(4) <= "0111100000111000";
num_sprite_ram(11)(5) <= "0011000001110000";
num_sprite_ram(11)(6) <= "0000000011100000";
num_sprite_ram(11)(7) <= "0000000111000000";
num_sprite_ram(11)(8) <= "0000001110000000";
num_sprite_ram(11)(9) <= "0000011100000000";
num_sprite_ram(11)(10) <="0000111000001100";
num_sprite_ram(11)(11) <="0001110000011110";
num_sprite_ram(11)(12) <="0011100000110011";
num_sprite_ram(11)(13) <="0111000000110011";
num_sprite_ram(11)(14) <="1110000000011110";
num_sprite_ram(11)(15) <="0000000000001100";

char_sprite_ram(0)(0) <= "0000000000000000";
char_sprite_ram(0)(1) <= "0001111111111000";
char_sprite_ram(0)(2) <= "0011111111111100";
char_sprite_ram(0)(3) <= "0111111001111110";
char_sprite_ram(0)(4) <= "0111110000111110";
char_sprite_ram(0)(5) <= "0111110000111100";
char_sprite_ram(0)(6) <= "0111111001111000";
char_sprite_ram(0)(7) <= "0111111111110000";
char_sprite_ram(0)(8) <= "0111111111111000";
char_sprite_ram(0)(9) <= "0111100000111100";
char_sprite_ram(0)(10) <="0111100000011110";
char_sprite_ram(0)(11) <="0111100000011110";
char_sprite_ram(0)(12) <="0111100000011110";
char_sprite_ram(0)(13) <="0111100000011110";
char_sprite_ram(0)(14) <="0111100000011110";
char_sprite_ram(0)(15) <="0000000000000000";

char_sprite_ram(1)(0) <= "0000000000000000";
char_sprite_ram(1)(1) <= "0011111111111110";
char_sprite_ram(1)(2) <= "0111111111111110";
char_sprite_ram(1)(3) <= "0111100000000000";
char_sprite_ram(1)(4) <= "0111100000000000";
char_sprite_ram(1)(5) <= "0111100000000000";
char_sprite_ram(1)(6) <= "0111100000000000";
char_sprite_ram(1)(7) <= "0111111111111110";
char_sprite_ram(1)(8) <= "0111111111111110";
char_sprite_ram(1)(9) <= "0111100000000000";
char_sprite_ram(1)(10) <="0111100000000000";
char_sprite_ram(1)(11) <="0111100000000000";
char_sprite_ram(1)(12) <="0111100000000000";
char_sprite_ram(1)(13) <="0111111111111110";
char_sprite_ram(1)(14) <="0011111111111110";
char_sprite_ram(1)(15) <="0000000000000000";

char_sprite_ram(2)(0) <= "0000000000000000";
char_sprite_ram(2)(1) <= "0001111111111100";
char_sprite_ram(2)(2) <= "0011111111111110";
char_sprite_ram(2)(3) <= "0111111000000000";
char_sprite_ram(2)(4) <= "0111110000000000";
char_sprite_ram(2)(5) <= "0111100000000000";
char_sprite_ram(2)(6) <= "0111100000000000";
char_sprite_ram(2)(7) <= "0111100000000000";
char_sprite_ram(2)(8) <= "0111100000000000";
char_sprite_ram(2)(9) <= "0111100000000000";
char_sprite_ram(2)(10) <="0111100000000000";
char_sprite_ram(2)(11) <="0111110000000000";
char_sprite_ram(2)(12) <="0111111000000000";
char_sprite_ram(2)(13) <="0011111111111110";
char_sprite_ram(2)(14) <="0001111111111100";
char_sprite_ram(2)(15) <="0000000000000000";

char_sprite_ram(3)(0) <= "0000000000000000";
char_sprite_ram(3)(1) <= "0001111111111000";
char_sprite_ram(3)(2) <= "0011111111111100";
char_sprite_ram(3)(3) <= "0111111001111110";
char_sprite_ram(3)(4) <= "0111110000111110";
char_sprite_ram(3)(5) <= "0111100000011110";
char_sprite_ram(3)(6) <= "0111100000011110";
char_sprite_ram(3)(7) <= "0111100000011110";
char_sprite_ram(3)(8) <= "0111100000011110";
char_sprite_ram(3)(9) <= "0111100000011110";
char_sprite_ram(3)(10) <="0111100000011110";
char_sprite_ram(3)(11) <="0111110000111110";
char_sprite_ram(3)(12) <="0111111001111110";
char_sprite_ram(3)(13) <="0011111111111100";
char_sprite_ram(3)(14) <="0001111111111000";
char_sprite_ram(3)(15) <="0000000000000000";

char_sprite_ram(4)(0) <= "0000000000000000";
char_sprite_ram(4)(1) <= "0111111111111000";
char_sprite_ram(4)(2) <= "0111111111111100";
char_sprite_ram(4)(3) <= "0111100001111110";
```

```
        char_sprite_ram(4)(4) <= "0111100000111110";
        char_sprite_ram(4)(5) <= "0111100000011110";
        char_sprite_ram(4)(6) <= "0111100000011110";
        char_sprite_ram(4)(7) <= "0111100000011110";
        char_sprite_ram(4)(8) <= "0111100000011110";
        char_sprite_ram(4)(9) <= "0111100000011110";
        char_sprite_ram(4)(10) <="0111100000011110";
        char_sprite_ram(4)(11) <="0111100000111110";
        char_sprite_ram(4)(12) <="0111100001111110";
        char_sprite_ram(4)(13) <="0111111111111100";
        char_sprite_ram(4)(14) <="0111111111111000";
        char_sprite_ram(4)(15) <="0000000000000000";

        color_ram(0) <= "000000000000000000000000000000";
        color_ram(1) <= "111111111111111111111000000000";
        color_ram(2) <= "111111111100000000000000000000";
        color_ram(3) <= "000000000010000000000000000000";
        color_ram(4) <= "111111111110000000000100000000";
        color_ram(5) <= "100000000000000000000000000000";
        color_ram(6) <= "111111111111111111111111111111";


pacman_sprite_0_ram(0)(0)<="110";pacman_sprite_0_ram(0)(1)<="110";pacman_sprite
_0_ram(0)(2)<="110";pacman_sprite_0_ram(0)(3)<="110";pacman_sprite_0_ram(0)(4)<
="110";pacman_sprite_0_ram(0)(5)<="110";pacman_sprite_0_ram(0)(6)<="110";pacma
n_sprite_0_ram(0)(7)<="110";pacman_sprite_0_ram(0)(8)<="110";pacman_sprite_0_ra
m(0)(9)<="110";pacman_sprite_0_ram(0)(10)<="110";pacman_sprite_0_ram(0)(11)<="
110";pacman_sprite_0_ram(0)(12)<="110";pacman_sprite_0_ram(0)(13)<="110";pacma
n_sprite_0_ram(0)(14)<="110";pacman_sprite_0_ram(0)(15)<="110";

pacman_sprite_0_ram(1)(0)<="110";pacman_sprite_0_ram(1)(1)<="011";pacman_sprite
_0_ram(1)(2)<="011";pacman_sprite_0_ram(1)(3)<="011";pacman_sprite_0_ram(1)(4)<
="011";pacman_sprite_0_ram(1)(5)<="011";pacman_sprite_0_ram(1)(6)<="001";pacma
n_sprite_0_ram(1)(7)<="001";pacman_sprite_0_ram(1)(8)<="001";pacman_sprite_0_ra
m(1)(9)<="001";pacman_sprite_0_ram(1)(10)<="011";pacman_sprite_0_ram(1)(11)<="
011";pacman_sprite_0_ram(1)(12)<="011";pacman_sprite_0_ram(1)(13)<="011";pacma
n_sprite_0_ram(1)(14)<="011";pacman_sprite_0_ram(1)(15)<="011";

pacman_sprite_0_ram(2)(0)<="110";pacman_sprite_0_ram(2)(1)<="011";pacman_sprite
_0_ram(2)(2)<="011";pacman_sprite_0_ram(2)(3)<="011";pacman_sprite_0_ram(2)(4)<
```

```
="001";pacman_sprite_0_ram(2)(5)<="001";pacman_sprite_0_ram(2)(6)<="001";pacma
n_sprite_0_ram(2)(7)<="001";pacman_sprite_0_ram(2)(8)<="001";pacman_sprite_0_ra
m(2)(9)<="001";pacman_sprite_0_ram(2)(10)<="001";pacman_sprite_0_ram(2)(11)<="
001";pacman_sprite_0_ram(2)(12)<="011";pacman_sprite_0_ram(2)(13)<="011";pacma
n_sprite_0_ram(2)(14)<="011";pacman_sprite_0_ram(2)(15)<="011";

pacman_sprite_0_ram(3)(0)<="110";pacman_sprite_0_ram(3)(1)<="011";pacman_sprite
_0_ram(3)(2)<="011";pacman_sprite_0_ram(3)(3)<="001";pacman_sprite_0_ram(3)(4)<
="001";pacman_sprite_0_ram(3)(5)<="001";pacman_sprite_0_ram(3)(6)<="001";pacma
n_sprite_0_ram(3)(7)<="001";pacman_sprite_0_ram(3)(8)<="001";pacman_sprite_0_ra
m(3)(9)<="001";pacman_sprite_0_ram(3)(10)<="001";pacman_sprite_0_ram(3)(11)<="
001";pacman_sprite_0_ram(3)(12)<="001";pacman_sprite_0_ram(3)(13)<="011";pacma
n_sprite_0_ram(3)(14)<="011";pacman_sprite_0_ram(3)(15)<="011";

pacman_sprite_0_ram(4)(0)<="110";pacman_sprite_0_ram(4)(1)<="011";pacman_sprite
_0_ram(4)(2)<="001";pacman_sprite_0_ram(4)(3)<="001";pacman_sprite_0_ram(4)(4)<
="001";pacman_sprite_0_ram(4)(5)<="001";pacman_sprite_0_ram(4)(6)<="001";pacma
n_sprite_0_ram(4)(7)<="001";pacman_sprite_0_ram(4)(8)<="001";pacman_sprite_0_ra
m(4)(9)<="001";pacman_sprite_0_ram(4)(10)<="001";pacman_sprite_0_ram(4)(11)<="
001";pacman_sprite_0_ram(4)(12)<="001";pacman_sprite_0_ram(4)(13)<="001";pacma
n_sprite_0_ram(4)(14)<="011";pacman_sprite_0_ram(4)(15)<="011";

pacman_sprite_0_ram(5)(0)<="110";pacman_sprite_0_ram(5)(1)<="001";pacman_sprite
_0_ram(5)(2)<="001";pacman_sprite_0_ram(5)(3)<="001";pacman_sprite_0_ram(5)(4)<
="001";pacman_sprite_0_ram(5)(5)<="001";pacman_sprite_0_ram(5)(6)<="001";pacma
n_sprite_0_ram(5)(7)<="001";pacman_sprite_0_ram(5)(8)<="001";pacman_sprite_0_ra
m(5)(9)<="001";pacman_sprite_0_ram(5)(10)<="001";pacman_sprite_0_ram(5)(11)<="
001";pacman_sprite_0_ram(5)(12)<="001";pacman_sprite_0_ram(5)(13)<="001";pacma
n_sprite_0_ram(5)(14)<="001";pacman_sprite_0_ram(5)(15)<="011";

pacman_sprite_0_ram(6)(0)<="110";pacman_sprite_0_ram(6)(1)<="001";pacman_sprite
_0_ram(6)(2)<="001";pacman_sprite_0_ram(6)(3)<="001";pacman_sprite_0_ram(6)(4)<
="001";pacman_sprite_0_ram(6)(5)<="001";pacman_sprite_0_ram(6)(6)<="001";pacma
n_sprite_0_ram(6)(7)<="001";pacman_sprite_0_ram(6)(8)<="001";pacman_sprite_0_ra
m(6)(9)<="001";pacman_sprite_0_ram(6)(10)<="001";pacman_sprite_0_ram(6)(11)<="
001";pacman_sprite_0_ram(6)(12)<="001";pacman_sprite_0_ram(6)(13)<="001";pacma
n_sprite_0_ram(6)(14)<="001";pacman_sprite_0_ram(6)(15)<="011";

pacman_sprite_0_ram(7)(0)<="110";pacman_sprite_0_ram(7)(1)<="001";pacman_sprite
_0_ram(7)(2)<="001";pacman_sprite_0_ram(7)(3)<="001";pacman_sprite_0_ram(7)(4)<
```

="001";pacman_sprite_0_ram(7)(5)<="001";pacman_sprite_0_ram(7)(6)<="001";pacma
n_sprite_0_ram(7)(7)<="001";pacman_sprite_0_ram(7)(8)<="001";pacman_sprite_0_ra
m(7)(9)<="001";pacman_sprite_0_ram(7)(10)<="001";pacman_sprite_0_ram(7)(11)<="
001";pacman_sprite_0_ram(7)(12)<="001";pacman_sprite_0_ram(7)(13)<="001";pacma
n_sprite_0_ram(7)(14)<="001";pacman_sprite_0_ram(7)(15)<="011";

pacman_sprite_0_ram(8)(0)<="110";pacman_sprite_0_ram(8)(1)<="001";pacman_sprite
_0_ram(8)(2)<="001";pacman_sprite_0_ram(8)(3)<="001";pacman_sprite_0_ram(8)(4)<
="001";pacman_sprite_0_ram(8)(5)<="001";pacman_sprite_0_ram(8)(6)<="001";pacma
n_sprite_0_ram(8)(7)<="001";pacman_sprite_0_ram(8)(8)<="001";pacman_sprite_0_ra
m(8)(9)<="001";pacman_sprite_0_ram(8)(10)<="001";pacman_sprite_0_ram(8)(11)<="
001";pacman_sprite_0_ram(8)(12)<="001";pacman_sprite_0_ram(8)(13)<="001";pacma
n_sprite_0_ram(8)(14)<="001";pacman_sprite_0_ram(8)(15)<="011";

pacman_sprite_0_ram(9)(0)<="110";pacman_sprite_0_ram(9)(1)<="001";pacman_sprite
_0_ram(9)(2)<="001";pacman_sprite_0_ram(9)(3)<="001";pacman_sprite_0_ram(9)(4)<
="001";pacman_sprite_0_ram(9)(5)<="001";pacman_sprite_0_ram(9)(6)<="001";pacma
n_sprite_0_ram(9)(7)<="001";pacman_sprite_0_ram(9)(8)<="001";pacman_sprite_0_ra
m(9)(9)<="001";pacman_sprite_0_ram(9)(10)<="001";pacman_sprite_0_ram(9)(11)<="
001";pacman_sprite_0_ram(9)(12)<="001";pacman_sprite_0_ram(9)(13)<="001";pacma
n_sprite_0_ram(9)(14)<="001";pacman_sprite_0_ram(9)(15)<="011";

pacman_sprite_0_ram(10)(0)<="110";pacman_sprite_0_ram(10)(1)<="001";pacman_spri
te_0_ram(10)(2)<="001";pacman_sprite_0_ram(10)(3)<="001";pacman_sprite_0_ram(10
)(4)<="001";pacman_sprite_0_ram(10)(5)<="001";pacman_sprite_0_ram(10)(6)<="001"
;pacman_sprite_0_ram(10)(7)<="001";pacman_sprite_0_ram(10)(8)<="001";pacman_spr
ite_0_ram(10)(9)<="001";pacman_sprite_0_ram(10)(10)<="001";pacman_sprite_0_ram(
10)(11)<="001";pacman_sprite_0_ram(10)(12)<="001";pacman_sprite_0_ram(10)(13)<
="001";pacman_sprite_0_ram(10)(14)<="001";pacman_sprite_0_ram(10)(15)<="011";

pacman_sprite_0_ram(11)(0)<="110";pacman_sprite_0_ram(11)(1)<="011";pacman_spri
te_0_ram(11)(2)<="001";pacman_sprite_0_ram(11)(3)<="001";pacman_sprite_0_ram(11
)(4)<="001";pacman_sprite_0_ram(11)(5)<="001";pacman_sprite_0_ram(11)(6)<="001"
;pacman_sprite_0_ram(11)(7)<="001";pacman_sprite_0_ram(11)(8)<="001";pacman_spr
ite_0_ram(11)(9)<="001";pacman_sprite_0_ram(11)(10)<="001";pacman_sprite_0_ram(
11)(11)<="001";pacman_sprite_0_ram(11)(12)<="001";pacman_sprite_0_ram(11)(13)<
="001";pacman_sprite_0_ram(11)(14)<="011";pacman_sprite_0_ram(11)(15)<="011";

pacman_sprite_0_ram(12)(0)<="110";pacman_sprite_0_ram(12)(1)<="011";pacman_spri
te_0_ram(12)(2)<="011";pacman_sprite_0_ram(12)(3)<="001";pacman_sprite_0_ram(12

```
)(4)<="001";pacman_sprite_0_ram(12)(5)<="001";pacman_sprite_0_ram(12)(6)<="001"
;pacman_sprite_0_ram(12)(7)<="001";pacman_sprite_0_ram(12)(8)<="001";pacman_spr
ite_0_ram(12)(9)<="001";pacman_sprite_0_ram(12)(10)<="001";pacman_sprite_0_ram(
12)(11)<="001";pacman_sprite_0_ram(12)(12)<="001";pacman_sprite_0_ram(12)(13)<
="011";pacman_sprite_0_ram(12)(14)<="011";pacman_sprite_0_ram(12)(15)<="011";

pacman_sprite_0_ram(13)(0)<="110";pacman_sprite_0_ram(13)(1)<="011";pacman_spri
te_0_ram(13)(2)<="011";pacman_sprite_0_ram(13)(3)<="011";pacman_sprite_0_ram(13
)(4)<="001";pacman_sprite_0_ram(13)(5)<="001";pacman_sprite_0_ram(13)(6)<="001"
;pacman_sprite_0_ram(13)(7)<="001";pacman_sprite_0_ram(13)(8)<="001";pacman_spr
ite_0_ram(13)(9)<="001";pacman_sprite_0_ram(13)(10)<="001";pacman_sprite_0_ram(
13)(11)<="001";pacman_sprite_0_ram(13)(12)<="011";pacman_sprite_0_ram(13)(13)<
="011";pacman_sprite_0_ram(13)(14)<="011";pacman_sprite_0_ram(13)(15)<="011";

pacman_sprite_0_ram(14)(0)<="110";pacman_sprite_0_ram(14)(1)<="011";pacman_spri
te_0_ram(14)(2)<="011";pacman_sprite_0_ram(14)(3)<="011";pacman_sprite_0_ram(14
)(4)<="011";pacman_sprite_0_ram(14)(5)<="011";pacman_sprite_0_ram(14)(6)<="001"
;pacman_sprite_0_ram(14)(7)<="001";pacman_sprite_0_ram(14)(8)<="001";pacman_spr
ite_0_ram(14)(9)<="001";pacman_sprite_0_ram(14)(10)<="011";pacman_sprite_0_ram(
14)(11)<="011";pacman_sprite_0_ram(14)(12)<="011";pacman_sprite_0_ram(14)(13)<
="011";pacman_sprite_0_ram(14)(14)<="011";pacman_sprite_0_ram(14)(15)<="011";

pacman_sprite_0_ram(15)(0)<="110";pacman_sprite_0_ram(15)(1)<="011";pacman_spri
te_0_ram(15)(2)<="011";pacman_sprite_0_ram(15)(3)<="011";pacman_sprite_0_ram(15
)(4)<="011";pacman_sprite_0_ram(15)(5)<="011";pacman_sprite_0_ram(15)(6)<="011"
;pacman_sprite_0_ram(15)(7)<="011";pacman_sprite_0_ram(15)(8)<="011";pacman_spr
ite_0_ram(15)(9)<="011";pacman_sprite_0_ram(15)(10)<="011";pacman_sprite_0_ram(
15)(11)<="011";pacman_sprite_0_ram(15)(12)<="011";pacman_sprite_0_ram(15)(13)<
="011";pacman_sprite_0_ram(15)(14)<="011";pacman_sprite_0_ram(15)(15)<="011";


pacman_sprite_right_ram(0)(0)<="110";pacman_sprite_right_ram(0)(1)<="110";pacman
_sprite_right_ram(0)(2)<="110";pacman_sprite_right_ram(0)(3)<="110";pacman_sprite_
right_ram(0)(4)<="110";pacman_sprite_right_ram(0)(5)<="110";pacman_sprite_right_ra
m(0)(6)<="110";pacman_sprite_right_ram(0)(7)<="110";pacman_sprite_right_ram(0)(8)
<="110";pacman_sprite_right_ram(0)(9)<="110";pacman_sprite_right_ram(0)(10)<="11
0";pacman_sprite_right_ram(0)(11)<="110";pacman_sprite_right_ram(0)(12)<="110";pa
cman_sprite_right_ram(0)(13)<="110";pacman_sprite_right_ram(0)(14)<="110";pacman
_sprite_right_ram(0)(15)<="011";
```

```
pacman_sprite_right_ram(1)(0)<="110";pacman_sprite_right_ram(1)(1)<="011";pacman
_sprite_right_ram(1)(2)<="011";pacman_sprite_right_ram(1)(3)<="011";pacman_sprite_
right_ram(1)(4)<="011";pacman_sprite_right_ram(1)(5)<="011";pacman_sprite_right_ra
m(1)(6)<="001";pacman_sprite_right_ram(1)(7)<="001";pacman_sprite_right_ram(1)(8)
<="001";pacman_sprite_right_ram(1)(9)<="001";pacman_sprite_right_ram(1)(10)<="01
1";pacman_sprite_right_ram(1)(11)<="011";pacman_sprite_right_ram(1)(12)<="011";pa
cman_sprite_right_ram(1)(13)<="011";pacman_sprite_right_ram(1)(14)<="011";pacman
_sprite_right_ram(1)(15)<="011";

pacman_sprite_right_ram(2)(0)<="110";pacman_sprite_right_ram(2)(1)<="011";pacman
_sprite_right_ram(2)(2)<="011";pacman_sprite_right_ram(2)(3)<="011";pacman_sprite_
right_ram(2)(4)<="001";pacman_sprite_right_ram(2)(5)<="001";pacman_sprite_right_ra
m(2)(6)<="001";pacman_sprite_right_ram(2)(7)<="001";pacman_sprite_right_ram(2)(8)
<="001";pacman_sprite_right_ram(2)(9)<="001";pacman_sprite_right_ram(2)(10)<="00
1";pacman_sprite_right_ram(2)(11)<="001";pacman_sprite_right_ram(2)(12)<="011";pa
cman_sprite_right_ram(2)(13)<="011";pacman_sprite_right_ram(2)(14)<="011";pacman
_sprite_right_ram(2)(15)<="011";

pacman_sprite_right_ram(3)(0)<="110";pacman_sprite_right_ram(3)(1)<="011";pacman
_sprite_right_ram(3)(2)<="011";pacman_sprite_right_ram(3)(3)<="001";pacman_sprite_
right_ram(3)(4)<="001";pacman_sprite_right_ram(3)(5)<="001";pacman_sprite_right_ra
m(3)(6)<="001";pacman_sprite_right_ram(3)(7)<="001";pacman_sprite_right_ram(3)(8)
<="001";pacman_sprite_right_ram(3)(9)<="001";pacman_sprite_right_ram(3)(10)<="00
1";pacman_sprite_right_ram(3)(11)<="001";pacman_sprite_right_ram(3)(12)<="001";pa
cman_sprite_right_ram(3)(13)<="011";pacman_sprite_right_ram(3)(14)<="011";pacman
_sprite_right_ram(3)(15)<="011";

pacman_sprite_right_ram(4)(0)<="110";pacman_sprite_right_ram(4)(1)<="011";pacman
_sprite_right_ram(4)(2)<="001";pacman_sprite_right_ram(4)(3)<="001";pacman_sprite_
right_ram(4)(4)<="001";pacman_sprite_right_ram(4)(5)<="001";pacman_sprite_right_ra
m(4)(6)<="001";pacman_sprite_right_ram(4)(7)<="001";pacman_sprite_right_ram(4)(8)
<="001";pacman_sprite_right_ram(4)(9)<="001";pacman_sprite_right_ram(4)(10)<="00
1";pacman_sprite_right_ram(4)(11)<="001";pacman_sprite_right_ram(4)(12)<="001";pa
cman_sprite_right_ram(4)(13)<="001";pacman_sprite_right_ram(4)(14)<="011";pacman
_sprite_right_ram(4)(15)<="011";

pacman_sprite_right_ram(5)(0)<="110";pacman_sprite_right_ram(5)(1)<="001";pacman
_sprite_right_ram(5)(2)<="001";pacman_sprite_right_ram(5)(3)<="001";pacman_sprite_
right_ram(5)(4)<="001";pacman_sprite_right_ram(5)(5)<="001";pacman_sprite_right_ra
m(5)(6)<="001";pacman_sprite_right_ram(5)(7)<="001";pacman_sprite_right_ram(5)(8)
```

```
<="001";pacman_sprite_right_ram(5)(9)<="001";pacman_sprite_right_ram(5)(10)<="00
1";pacman_sprite_right_ram(5)(11)<="011";pacman_sprite_right_ram(5)(12)<="011";pa
cman_sprite_right_ram(5)(13)<="011";pacman_sprite_right_ram(5)(14)<="011";pacman
_sprite_right_ram(5)(15)<="011";

pacman_sprite_right_ram(6)(0)<="110";pacman_sprite_right_ram(6)(1)<="001";pacman
_sprite_right_ram(6)(2)<="001";pacman_sprite_right_ram(6)(3)<="001";pacman_sprite_
right_ram(6)(4)<="001";pacman_sprite_right_ram(6)(5)<="001";pacman_sprite_right_ra
m(6)(6)<="001";pacman_sprite_right_ram(6)(7)<="001";pacman_sprite_right_ram(6)(8)
<="001";pacman_sprite_right_ram(6)(9)<="011";pacman_sprite_right_ram(6)(10)<="01
1";pacman_sprite_right_ram(6)(11)<="011";pacman_sprite_right_ram(6)(12)<="011";pa
cman_sprite_right_ram(6)(13)<="011";pacman_sprite_right_ram(6)(14)<="011";pacman
_sprite_right_ram(6)(15)<="011";

pacman_sprite_right_ram(7)(0)<="110";pacman_sprite_right_ram(7)(1)<="001";pacman
_sprite_right_ram(7)(2)<="001";pacman_sprite_right_ram(7)(3)<="001";pacman_sprite_
right_ram(7)(4)<="001";pacman_sprite_right_ram(7)(5)<="001";pacman_sprite_right_ra
m(7)(6)<="001";pacman_sprite_right_ram(7)(7)<="011";pacman_sprite_right_ram(7)(8)
<="011";pacman_sprite_right_ram(7)(9)<="011";pacman_sprite_right_ram(7)(10)<="01
1";pacman_sprite_right_ram(7)(11)<="011";pacman_sprite_right_ram(7)(12)<="011";pa
cman_sprite_right_ram(7)(13)<="011";pacman_sprite_right_ram(7)(14)<="011";pacman
_sprite_right_ram(7)(15)<="011";

pacman_sprite_right_ram(8)(0)<="110";pacman_sprite_right_ram(8)(1)<="001";pacman
_sprite_right_ram(8)(2)<="001";pacman_sprite_right_ram(8)(3)<="001";pacman_sprite_
right_ram(8)(4)<="001";pacman_sprite_right_ram(8)(5)<="001";pacman_sprite_right_ra
m(8)(6)<="001";pacman_sprite_right_ram(8)(7)<="001";pacman_sprite_right_ram(8)(8)
<="011";pacman_sprite_right_ram(8)(9)<="011";pacman_sprite_right_ram(8)(10)<="01
1";pacman_sprite_right_ram(8)(11)<="011";pacman_sprite_right_ram(8)(12)<="011";pa
cman_sprite_right_ram(8)(13)<="011";pacman_sprite_right_ram(8)(14)<="011";pacman
_sprite_right_ram(8)(15)<="011";

pacman_sprite_right_ram(9)(0)<="110";pacman_sprite_right_ram(9)(1)<="001";pacman
_sprite_right_ram(9)(2)<="001";pacman_sprite_right_ram(9)(3)<="001";pacman_sprite_
right_ram(9)(4)<="001";pacman_sprite_right_ram(9)(5)<="001";pacman_sprite_right_ra
m(9)(6)<="001";pacman_sprite_right_ram(9)(7)<="001";pacman_sprite_right_ram(9)(8)
<="001";pacman_sprite_right_ram(9)(9)<="001";pacman_sprite_right_ram(9)(10)<="01
1";pacman_sprite_right_ram(9)(11)<="011";pacman_sprite_right_ram(9)(12)<="011";pa
cman_sprite_right_ram(9)(13)<="011";pacman_sprite_right_ram(9)(14)<="011";pacman
_sprite_right_ram(9)(15)<="011";
```

```
pacman_sprite_right_ram(10)(0)<="110";pacman_sprite_right_ram(10)(1)<="001";pacm
an_sprite_right_ram(10)(2)<="001";pacman_sprite_right_ram(10)(3)<="001";pacman_sp
rite_right_ram(10)(4)<="001";pacman_sprite_right_ram(10)(5)<="001";pacman_sprite_r
ight_ram(10)(6)<="001";pacman_sprite_right_ram(10)(7)<="001";pacman_sprite_right_
ram(10)(8)<="001";pacman_sprite_right_ram(10)(9)<="001";pacman_sprite_right_ram(
10)(10)<="001";pacman_sprite_right_ram(10)(11)<="001";pacman_sprite_right_ram(10)
(12)<="011";pacman_sprite_right_ram(10)(13)<="011";pacman_sprite_right_ram(10)(14
)<="011";pacman_sprite_right_ram(10)(15)<="011";

pacman_sprite_right_ram(11)(0)<="110";pacman_sprite_right_ram(11)(1)<="011";pacm
an_sprite_right_ram(11)(2)<="001";pacman_sprite_right_ram(11)(3)<="001";pacman_sp
rite_right_ram(11)(4)<="001";pacman_sprite_right_ram(11)(5)<="001";pacman_sprite_r
ight_ram(11)(6)<="001";pacman_sprite_right_ram(11)(7)<="001";pacman_sprite_right_
ram(11)(8)<="001";pacman_sprite_right_ram(11)(9)<="001";pacman_sprite_right_ram(
11)(10)<="001";pacman_sprite_right_ram(11)(11)<="001";pacman_sprite_right_ram(11)
(12)<="001";pacman_sprite_right_ram(11)(13)<="001";pacman_sprite_right_ram(11)(14
)<="011";pacman_sprite_right_ram(11)(15)<="011";

pacman_sprite_right_ram(12)(0)<="110";pacman_sprite_right_ram(12)(1)<="011";pacm
an_sprite_right_ram(12)(2)<="011";pacman_sprite_right_ram(12)(3)<="001";pacman_sp
rite_right_ram(12)(4)<="001";pacman_sprite_right_ram(12)(5)<="001";pacman_sprite_r
ight_ram(12)(6)<="001";pacman_sprite_right_ram(12)(7)<="001";pacman_sprite_right_
ram(12)(8)<="001";pacman_sprite_right_ram(12)(9)<="001";pacman_sprite_right_ram(
12)(10)<="001";pacman_sprite_right_ram(12)(11)<="001";pacman_sprite_right_ram(12)
(12)<="001";pacman_sprite_right_ram(12)(13)<="011";pacman_sprite_right_ram(12)(14
)<="011";pacman_sprite_right_ram(12)(15)<="011";

pacman_sprite_right_ram(13)(0)<="110";pacman_sprite_right_ram(13)(1)<="011";pacm
an_sprite_right_ram(13)(2)<="011";pacman_sprite_right_ram(13)(3)<="011";pacman_sp
rite_right_ram(13)(4)<="001";pacman_sprite_right_ram(13)(5)<="001";pacman_sprite_r
ight_ram(13)(6)<="001";pacman_sprite_right_ram(13)(7)<="001";pacman_sprite_right_
ram(13)(8)<="001";pacman_sprite_right_ram(13)(9)<="001";pacman_sprite_right_ram(
13)(10)<="001";pacman_sprite_right_ram(13)(11)<="001";pacman_sprite_right_ram(13)
(12)<="011";pacman_sprite_right_ram(13)(13)<="011";pacman_sprite_right_ram(13)(14
)<="011";pacman_sprite_right_ram(13)(15)<="011";

pacman_sprite_right_ram(14)(0)<="110";pacman_sprite_right_ram(14)(1)<="011";pacm
an_sprite_right_ram(14)(2)<="011";pacman_sprite_right_ram(14)(3)<="011";pacman_sp
rite_right_ram(14)(4)<="011";pacman_sprite_right_ram(14)(5)<="011";pacman_sprite_r
```

```
ight_ram(14)(6)<="001";pacman_sprite_right_ram(14)(7)<="001";pacman_sprite_right_
ram(14)(8)<="001";pacman_sprite_right_ram(14)(9)<="001";pacman_sprite_right_ram(
14)(10)<="011";pacman_sprite_right_ram(14)(11)<="011";pacman_sprite_right_ram(14)
(12)<="011";pacman_sprite_right_ram(14)(13)<="011";pacman_sprite_right_ram(14)(14
)<="011";pacman_sprite_right_ram(14)(15)<="011";

pacman_sprite_right_ram(15)(0)<="110";pacman_sprite_right_ram(15)(1)<="011";pacm
an_sprite_right_ram(15)(2)<="011";pacman_sprite_right_ram(15)(3)<="011";pacman_sp
rite_right_ram(15)(4)<="011";pacman_sprite_right_ram(15)(5)<="011";pacman_sprite_r
ight_ram(15)(6)<="011";pacman_sprite_right_ram(15)(7)<="011";pacman_sprite_right_
ram(15)(8)<="011";pacman_sprite_right_ram(15)(9)<="011";pacman_sprite_right_ram(
15)(10)<="011";pacman_sprite_right_ram(15)(11)<="011";pacman_sprite_right_ram(15)
(12)<="011";pacman_sprite_right_ram(15)(13)<="011";pacman_sprite_right_ram(15)(14
)<="011";pacman_sprite_right_ram(15)(15)<="011";


pacman_sprite_down_ram(0)(0)<="110";pacman_sprite_down_ram(0)(1)<="110";pacma
n_sprite_down_ram(0)(2)<="110";pacman_sprite_down_ram(0)(3)<="110";pacman_spri
te_down_ram(0)(4)<="110";pacman_sprite_down_ram(0)(5)<="110";pacman_sprite_do
wn_ram(0)(6)<="110";pacman_sprite_down_ram(0)(7)<="110";pacman_sprite_down_ra
m(0)(8)<="110";pacman_sprite_down_ram(0)(9)<="110";pacman_sprite_down_ram(0)(
10)<="110";pacman_sprite_down_ram(0)(11)<="110";pacman_sprite_down_ram(0)(12)
<="110";pacman_sprite_down_ram(0)(13)<="110";pacman_sprite_down_ram(0)(14)<="
110";pacman_sprite_down_ram(0)(15)<="110";

pacman_sprite_down_ram(1)(0)<="110";pacman_sprite_down_ram(1)(1)<="011";pacma
n_sprite_down_ram(1)(2)<="011";pacman_sprite_down_ram(1)(3)<="011";pacman_spri
te_down_ram(1)(4)<="011";pacman_sprite_down_ram(1)(5)<="011";pacman_sprite_do
wn_ram(1)(6)<="001";pacman_sprite_down_ram(1)(7)<="001";pacman_sprite_down_ra
m(1)(8)<="001";pacman_sprite_down_ram(1)(9)<="001";pacman_sprite_down_ram(1)(
10)<="011";pacman_sprite_down_ram(1)(11)<="011";pacman_sprite_down_ram(1)(12)
<="011";pacman_sprite_down_ram(1)(13)<="011";pacman_sprite_down_ram(1)(14)<="
011";pacman_sprite_down_ram(1)(15)<="011";

pacman_sprite_down_ram(2)(0)<="110";pacman_sprite_down_ram(2)(1)<="011";pacma
n_sprite_down_ram(2)(2)<="011";pacman_sprite_down_ram(2)(3)<="011";pacman_spri
te_down_ram(2)(4)<="001";pacman_sprite_down_ram(2)(5)<="001";pacman_sprite_do
wn_ram(2)(6)<="001";pacman_sprite_down_ram(2)(7)<="001";pacman_sprite_down_ra
m(2)(8)<="001";pacman_sprite_down_ram(2)(9)<="001";pacman_sprite_down_ram(2)(
10)<="001";pacman_sprite_down_ram(2)(11)<="001";pacman_sprite_down_ram(2)(12)
```

```
<="011";pacman_sprite_down_ram(2)(13)<="011";pacman_sprite_down_ram(2)(14)<="
011";pacman_sprite_down_ram(2)(15)<="011";

pacman_sprite_down_ram(3)(0)<="110";pacman_sprite_down_ram(3)(1)<="011";pacma
n_sprite_down_ram(3)(2)<="011";pacman_sprite_down_ram(3)(3)<="001";pacman_spri
te_down_ram(3)(4)<="001";pacman_sprite_down_ram(3)(5)<="001";pacman_sprite_do
wn_ram(3)(6)<="001";pacman_sprite_down_ram(3)(7)<="001";pacman_sprite_down_ra
m(3)(8)<="001";pacman_sprite_down_ram(3)(9)<="001";pacman_sprite_down_ram(3)(
10)<="001";pacman_sprite_down_ram(3)(11)<="001";pacman_sprite_down_ram(3)(12)
<="001";pacman_sprite_down_ram(3)(13)<="011";pacman_sprite_down_ram(3)(14)<="
011";pacman_sprite_down_ram(3)(15)<="011";

pacman_sprite_down_ram(4)(0)<="110";pacman_sprite_down_ram(4)(1)<="011";pacma
n_sprite_down_ram(4)(2)<="001";pacman_sprite_down_ram(4)(3)<="001";pacman_spri
te_down_ram(4)(4)<="001";pacman_sprite_down_ram(4)(5)<="001";pacman_sprite_do
wn_ram(4)(6)<="001";pacman_sprite_down_ram(4)(7)<="001";pacman_sprite_down_ra
m(4)(8)<="001";pacman_sprite_down_ram(4)(9)<="001";pacman_sprite_down_ram(4)(
10)<="001";pacman_sprite_down_ram(4)(11)<="001";pacman_sprite_down_ram(4)(12)
<="001";pacman_sprite_down_ram(4)(13)<="001";pacman_sprite_down_ram(4)(14)<="
011";pacman_sprite_down_ram(4)(15)<="011";

pacman_sprite_down_ram(5)(0)<="110";pacman_sprite_down_ram(5)(1)<="001";pacma
n_sprite_down_ram(5)(2)<="001";pacman_sprite_down_ram(5)(3)<="001";pacman_spri
te_down_ram(5)(4)<="001";pacman_sprite_down_ram(5)(5)<="001";pacman_sprite_do
wn_ram(5)(6)<="001";pacman_sprite_down_ram(5)(7)<="001";pacman_sprite_down_ra
m(5)(8)<="001";pacman_sprite_down_ram(5)(9)<="001";pacman_sprite_down_ram(5)(
10)<="001";pacman_sprite_down_ram(5)(11)<="001";pacman_sprite_down_ram(5)(12)
<="001";pacman_sprite_down_ram(5)(13)<="001";pacman_sprite_down_ram(5)(14)<="
001";pacman_sprite_down_ram(5)(15)<="011";

pacman_sprite_down_ram(6)(0)<="110";pacman_sprite_down_ram(6)(1)<="001";pacma
n_sprite_down_ram(6)(2)<="001";pacman_sprite_down_ram(6)(3)<="001";pacman_spri
te_down_ram(6)(4)<="001";pacman_sprite_down_ram(6)(5)<="001";pacman_sprite_do
wn_ram(6)(6)<="001";pacman_sprite_down_ram(6)(7)<="001";pacman_sprite_down_ra
m(6)(8)<="001";pacman_sprite_down_ram(6)(9)<="001";pacman_sprite_down_ram(6)(
10)<="001";pacman_sprite_down_ram(6)(11)<="001";pacman_sprite_down_ram(6)(12)
<="001";pacman_sprite_down_ram(6)(13)<="001";pacman_sprite_down_ram(6)(14)<="
001";pacman_sprite_down_ram(6)(15)<="011";

pacman_sprite_down_ram(7)(0)<="110";pacman_sprite_down_ram(7)(1)<="001";pacma
```

```
n_sprite_down_ram(7)(2)<="001";pacman_sprite_down_ram(7)(3)<="001";pacman_spri
te_down_ram(7)(4)<="001";pacman_sprite_down_ram(7)(5)<="001";pacman_sprite_do
wn_ram(7)(6)<="001";pacman_sprite_down_ram(7)(7)<="011";pacman_sprite_down_ra
m(7)(8)<="001";pacman_sprite_down_ram(7)(9)<="001";pacman_sprite_down_ram(7)(
10)<="001";pacman_sprite_down_ram(7)(11)<="001";pacman_sprite_down_ram(7)(12)
<="001";pacman_sprite_down_ram(7)(13)<="001";pacman_sprite_down_ram(7)(14)<="
001";pacman_sprite_down_ram(7)(15)<="011";

pacman_sprite_down_ram(8)(0)<="110";pacman_sprite_down_ram(8)(1)<="001";pacma
n_sprite_down_ram(8)(2)<="001";pacman_sprite_down_ram(8)(3)<="001";pacman_spri
te_down_ram(8)(4)<="001";pacman_sprite_down_ram(8)(5)<="001";pacman_sprite_do
wn_ram(8)(6)<="001";pacman_sprite_down_ram(8)(7)<="011";pacman_sprite_down_ra
m(8)(8)<="011";pacman_sprite_down_ram(8)(9)<="001";pacman_sprite_down_ram(8)(
10)<="001";pacman_sprite_down_ram(8)(11)<="001";pacman_sprite_down_ram(8)(12)
<="001";pacman_sprite_down_ram(8)(13)<="001";pacman_sprite_down_ram(8)(14)<="
001";pacman_sprite_down_ram(8)(15)<="011";

pacman_sprite_down_ram(9)(0)<="110";pacman_sprite_down_ram(9)(1)<="001";pacma
n_sprite_down_ram(9)(2)<="001";pacman_sprite_down_ram(9)(3)<="001";pacman_spri
te_down_ram(9)(4)<="001";pacman_sprite_down_ram(9)(5)<="001";pacman_sprite_do
wn_ram(9)(6)<="011";pacman_sprite_down_ram(9)(7)<="011";pacman_sprite_down_ra
m(9)(8)<="011";pacman_sprite_down_ram(9)(9)<="001";pacman_sprite_down_ram(9)(
10)<="001";pacman_sprite_down_ram(9)(11)<="001";pacman_sprite_down_ram(9)(12)
<="001";pacman_sprite_down_ram(9)(13)<="001";pacman_sprite_down_ram(9)(14)<="
001";pacman_sprite_down_ram(9)(15)<="011";

pacman_sprite_down_ram(10)(0)<="110";pacman_sprite_down_ram(10)(1)<="001";pac
man_sprite_down_ram(10)(2)<="001";pacman_sprite_down_ram(10)(3)<="001";pacma
n_sprite_down_ram(10)(4)<="001";pacman_sprite_down_ram(10)(5)<="001";pacman_s
prite_down_ram(10)(6)<="011";pacman_sprite_down_ram(10)(7)<="011";pacman_sprit
e_down_ram(10)(8)<="011";pacman_sprite_down_ram(10)(9)<="011";pacman_sprite_d
own_ram(10)(10)<="001";pacman_sprite_down_ram(10)(11)<="001";pacman_sprite_do
wn_ram(10)(12)<="001";pacman_sprite_down_ram(10)(13)<="001";pacman_sprite_do
wn_ram(10)(14)<="001";pacman_sprite_down_ram(10)(15)<="011";

pacman_sprite_down_ram(11)(0)<="110";pacman_sprite_down_ram(11)(1)<="011";pac
man_sprite_down_ram(11)(2)<="001";pacman_sprite_down_ram(11)(3)<="001";pacma
n_sprite_down_ram(11)(4)<="001";pacman_sprite_down_ram(11)(5)<="011";pacman_s
prite_down_ram(11)(6)<="011";pacman_sprite_down_ram(11)(7)<="011";pacman_sprit
e_down_ram(11)(8)<="011";pacman_sprite_down_ram(11)(9)<="011";pacman_sprite_d
```

```
own_ram(11)(10)<="001";pacman_sprite_down_ram(11)(11)<="001";pacman_sprite_do
wn_ram(11)(12)<="001";pacman_sprite_down_ram(11)(13)<="001";pacman_sprite_do
wn_ram(11)(14)<="011";pacman_sprite_down_ram(11)(15)<="011";

pacman_sprite_down_ram(12)(0)<="110";pacman_sprite_down_ram(12)(1)<="011";pac
man_sprite_down_ram(12)(2)<="011";pacman_sprite_down_ram(12)(3)<="001";pacma
n_sprite_down_ram(12)(4)<="001";pacman_sprite_down_ram(12)(5)<="011";pacman_s
prite_down_ram(12)(6)<="011";pacman_sprite_down_ram(12)(7)<="011";pacman_sprit
e_down_ram(12)(8)<="011";pacman_sprite_down_ram(12)(9)<="011";pacman_sprite_d
own_ram(12)(10)<="011";pacman_sprite_down_ram(12)(11)<="001";pacman_sprite_do
wn_ram(12)(12)<="001";pacman_sprite_down_ram(12)(13)<="011";pacman_sprite_do
wn_ram(12)(14)<="011";pacman_sprite_down_ram(12)(15)<="011";

pacman_sprite_down_ram(13)(0)<="110";pacman_sprite_down_ram(13)(1)<="011";pac
man_sprite_down_ram(13)(2)<="011";pacman_sprite_down_ram(13)(3)<="011";pacma
n_sprite_down_ram(13)(4)<="011";pacman_sprite_down_ram(13)(5)<="011";pacman_s
prite_down_ram(13)(6)<="011";pacman_sprite_down_ram(13)(7)<="011";pacman_sprit
e_down_ram(13)(8)<="011";pacman_sprite_down_ram(13)(9)<="011";pacman_sprite_d
own_ram(13)(10)<="011";pacman_sprite_down_ram(13)(11)<="001";pacman_sprite_do
wn_ram(13)(12)<="011";pacman_sprite_down_ram(13)(13)<="011";pacman_sprite_do
wn_ram(13)(14)<="011";pacman_sprite_down_ram(13)(15)<="011";

pacman_sprite_down_ram(14)(0)<="110";pacman_sprite_down_ram(14)(1)<="011";pac
man_sprite_down_ram(14)(2)<="011";pacman_sprite_down_ram(14)(3)<="011";pacma
n_sprite_down_ram(14)(4)<="011";pacman_sprite_down_ram(14)(5)<="011";pacman_s
prite_down_ram(14)(6)<="011";pacman_sprite_down_ram(14)(7)<="011";pacman_sprit
e_down_ram(14)(8)<="011";pacman_sprite_down_ram(14)(9)<="011";pacman_sprite_d
own_ram(14)(10)<="011";pacman_sprite_down_ram(14)(11)<="001";pacman_sprite_do
wn_ram(14)(12)<="011";pacman_sprite_down_ram(14)(13)<="011";pacman_sprite_do
wn_ram(14)(14)<="011";pacman_sprite_down_ram(14)(15)<="011";

pacman_sprite_down_ram(15)(0)<="110";pacman_sprite_down_ram(15)(1)<="011";pac
man_sprite_down_ram(15)(2)<="011";pacman_sprite_down_ram(15)(3)<="011";pacma
n_sprite_down_ram(15)(4)<="011";pacman_sprite_down_ram(15)(5)<="011";pacman_s
prite_down_ram(15)(6)<="011";pacman_sprite_down_ram(15)(7)<="011";pacman_sprit
e_down_ram(15)(8)<="011";pacman_sprite_down_ram(15)(9)<="011";pacman_sprite_d
own_ram(15)(10)<="011";pacman_sprite_down_ram(15)(11)<="011";pacman_sprite_do
wn_ram(15)(12)<="011";pacman_sprite_down_ram(15)(13)<="011";pacman_sprite_do
wn_ram(15)(14)<="011";pacman_sprite_down_ram(15)(15)<="011";
```

```
pacman_sprite_up_ram(0)(0)<="110";pacman_sprite_up_ram(0)(1)<="110";pacman_spri
te_up_ram(0)(2)<="110";pacman_sprite_up_ram(0)(3)<="110";pacman_sprite_up_ram(0
)(4)<="110";pacman_sprite_up_ram(0)(5)<="110";pacman_sprite_up_ram(0)(6)<="110"
;pacman_sprite_up_ram(0)(7)<="110";pacman_sprite_up_ram(0)(8)<="110";pacman_spr
ite_up_ram(0)(9)<="110";pacman_sprite_up_ram(0)(10)<="110";pacman_sprite_up_ram
(0)(11)<="110";pacman_sprite_up_ram(0)(12)<="110";pacman_sprite_up_ram(0)(13)<=
"110";pacman_sprite_up_ram(0)(14)<="110";pacman_sprite_up_ram(0)(15)<="110";

pacman_sprite_up_ram(1)(0)<="110";pacman_sprite_up_ram(1)(1)<="011";pacman_spri
te_up_ram(1)(2)<="011";pacman_sprite_up_ram(1)(3)<="011";pacman_sprite_up_ram(1
)(4)<="011";pacman_sprite_up_ram(1)(5)<="011";pacman_sprite_up_ram(1)(6)<="011"
;pacman_sprite_up_ram(1)(7)<="011";pacman_sprite_up_ram(1)(8)<="001";pacman_spr
ite_up_ram(1)(9)<="011";pacman_sprite_up_ram(1)(10)<="011";pacman_sprite_up_ram
(1)(11)<="011";pacman_sprite_up_ram(1)(12)<="011";pacman_sprite_up_ram(1)(13)<=
"011";pacman_sprite_up_ram(1)(14)<="011";pacman_sprite_up_ram(1)(15)<="011";

pacman_sprite_up_ram(2)(0)<="110";pacman_sprite_up_ram(2)(1)<="011";pacman_spri
te_up_ram(2)(2)<="011";pacman_sprite_up_ram(2)(3)<="011";pacman_sprite_up_ram(2
)(4)<="011";pacman_sprite_up_ram(2)(5)<="011";pacman_sprite_up_ram(2)(6)<="011"
;pacman_sprite_up_ram(2)(7)<="011";pacman_sprite_up_ram(2)(8)<="001";pacman_spr
ite_up_ram(2)(9)<="011";pacman_sprite_up_ram(2)(10)<="011";pacman_sprite_up_ram
(2)(11)<="001";pacman_sprite_up_ram(2)(12)<="011";pacman_sprite_up_ram(2)(13)<=
"011";pacman_sprite_up_ram(2)(14)<="011";pacman_sprite_up_ram(2)(15)<="011";

pacman_sprite_up_ram(3)(0)<="110";pacman_sprite_up_ram(3)(1)<="011";pacman_spri
te_up_ram(3)(2)<="011";pacman_sprite_up_ram(3)(3)<="001";pacman_sprite_up_ram(3
)(4)<="001";pacman_sprite_up_ram(3)(5)<="011";pacman_sprite_up_ram(3)(6)<="011"
;pacman_sprite_up_ram(3)(7)<="011";pacman_sprite_up_ram(3)(8)<="001";pacman_spr
ite_up_ram(3)(9)<="011";pacman_sprite_up_ram(3)(10)<="001";pacman_sprite_up_ram
(3)(11)<="001";pacman_sprite_up_ram(3)(12)<="001";pacman_sprite_up_ram(3)(13)<=
"011";pacman_sprite_up_ram(3)(14)<="011";pacman_sprite_up_ram(3)(15)<="011";

pacman_sprite_up_ram(4)(0)<="110";pacman_sprite_up_ram(4)(1)<="011";pacman_spri
te_up_ram(4)(2)<="001";pacman_sprite_up_ram(4)(3)<="001";pacman_sprite_up_ram(4
)(4)<="001";pacman_sprite_up_ram(4)(5)<="001";pacman_sprite_up_ram(4)(6)<="011"
;pacman_sprite_up_ram(4)(7)<="011";pacman_sprite_up_ram(4)(8)<="011";pacman_spr
ite_up_ram(4)(9)<="011";pacman_sprite_up_ram(4)(10)<="001";pacman_sprite_up_ram
(4)(11)<="001";pacman_sprite_up_ram(4)(12)<="001";pacman_sprite_up_ram(4)(13)<=
"001";pacman_sprite_up_ram(4)(14)<="011";pacman_sprite_up_ram(4)(15)<="011";
```

```
pacman_sprite_up_ram(5)(0)<="110";pacman_sprite_up_ram(5)(1)<="001";pacman_spri
te_up_ram(5)(2)<="001";pacman_sprite_up_ram(5)(3)<="001";pacman_sprite_up_ram(5
)(4)<="001";pacman_sprite_up_ram(5)(5)<="001";pacman_sprite_up_ram(5)(6)<="011"
;pacman_sprite_up_ram(5)(7)<="011";pacman_sprite_up_ram(5)(8)<="011";pacman_spr
ite_up_ram(5)(9)<="001";pacman_sprite_up_ram(5)(10)<="001";pacman_sprite_up_ram
(5)(11)<="001";pacman_sprite_up_ram(5)(12)<="001";pacman_sprite_up_ram(5)(13)<=
"001";pacman_sprite_up_ram(5)(14)<="001";pacman_sprite_up_ram(5)(15)<="011";

pacman_sprite_up_ram(6)(0)<="110";pacman_sprite_up_ram(6)(1)<="001";pacman_spri
te_up_ram(6)(2)<="001";pacman_sprite_up_ram(6)(3)<="001";pacman_sprite_up_ram(6
)(4)<="001";pacman_sprite_up_ram(6)(5)<="001";pacman_sprite_up_ram(6)(6)<="001"
;pacman_sprite_up_ram(6)(7)<="011";pacman_sprite_up_ram(6)(8)<="011";pacman_spr
ite_up_ram(6)(9)<="001";pacman_sprite_up_ram(6)(10)<="001";pacman_sprite_up_ram
(6)(11)<="001";pacman_sprite_up_ram(6)(12)<="001";pacman_sprite_up_ram(6)(13)<=
"001";pacman_sprite_up_ram(6)(14)<="001";pacman_sprite_up_ram(6)(15)<="011";

pacman_sprite_up_ram(7)(0)<="110";pacman_sprite_up_ram(7)(1)<="001";pacman_spri
te_up_ram(7)(2)<="001";pacman_sprite_up_ram(7)(3)<="001";pacman_sprite_up_ram(7
)(4)<="001";pacman_sprite_up_ram(7)(5)<="001";pacman_sprite_up_ram(7)(6)<="001"
;pacman_sprite_up_ram(7)(7)<="011";pacman_sprite_up_ram(7)(8)<="001";pacman_spr
ite_up_ram(7)(9)<="001";pacman_sprite_up_ram(7)(10)<="001";pacman_sprite_up_ram
(7)(11)<="001";pacman_sprite_up_ram(7)(12)<="001";pacman_sprite_up_ram(7)(13)<=
"001";pacman_sprite_up_ram(7)(14)<="001";pacman_sprite_up_ram(7)(15)<="011";

pacman_sprite_up_ram(8)(0)<="110";pacman_sprite_up_ram(8)(1)<="001";pacman_spri
te_up_ram(8)(2)<="001";pacman_sprite_up_ram(8)(3)<="001";pacman_sprite_up_ram(8
)(4)<="001";pacman_sprite_up_ram(8)(5)<="001";pacman_sprite_up_ram(8)(6)<="001"
;pacman_sprite_up_ram(8)(7)<="001";pacman_sprite_up_ram(8)(8)<="001";pacman_spr
ite_up_ram(8)(9)<="001";pacman_sprite_up_ram(8)(10)<="001";pacman_sprite_up_ram
(8)(11)<="001";pacman_sprite_up_ram(8)(12)<="001";pacman_sprite_up_ram(8)(13)<=
"001";pacman_sprite_up_ram(8)(14)<="001";pacman_sprite_up_ram(8)(15)<="011";

pacman_sprite_up_ram(9)(0)<="110";pacman_sprite_up_ram(9)(1)<="001";pacman_spri
te_up_ram(9)(2)<="001";pacman_sprite_up_ram(9)(3)<="001";pacman_sprite_up_ram(9
)(4)<="001";pacman_sprite_up_ram(9)(5)<="001";pacman_sprite_up_ram(9)(6)<="001"
;pacman_sprite_up_ram(9)(7)<="001";pacman_sprite_up_ram(9)(8)<="001";pacman_spr
ite_up_ram(9)(9)<="001";pacman_sprite_up_ram(9)(10)<="001";pacman_sprite_up_ram
(9)(11)<="001";pacman_sprite_up_ram(9)(12)<="001";pacman_sprite_up_ram(9)(13)<=
"001";pacman_sprite_up_ram(9)(14)<="001";pacman_sprite_up_ram(9)(15)<="011";
```

```
pacman_sprite_up_ram(10)(0)<="110";pacman_sprite_up_ram(10)(1)<="001";pacman_s
prite_up_ram(10)(2)<="001";pacman_sprite_up_ram(10)(3)<="001";pacman_sprite_up_
ram(10)(4)<="001";pacman_sprite_up_ram(10)(5)<="001";pacman_sprite_up_ram(10)(6
)<="001";pacman_sprite_up_ram(10)(7)<="001";pacman_sprite_up_ram(10)(8)<="001";
pacman_sprite_up_ram(10)(9)<="001";pacman_sprite_up_ram(10)(10)<="001";pacman_
sprite_up_ram(10)(11)<="001";pacman_sprite_up_ram(10)(12)<="001";pacman_sprite_
up_ram(10)(13)<="001";pacman_sprite_up_ram(10)(14)<="001";pacman_sprite_up_ram
(10)(15)<="011";

pacman_sprite_up_ram(11)(0)<="110";pacman_sprite_up_ram(11)(1)<="011";pacman_s
prite_up_ram(11)(2)<="001";pacman_sprite_up_ram(11)(3)<="001";pacman_sprite_up_
ram(11)(4)<="001";pacman_sprite_up_ram(11)(5)<="001";pacman_sprite_up_ram(11)(6
)<="001";pacman_sprite_up_ram(11)(7)<="001";pacman_sprite_up_ram(11)(8)<="001";
pacman_sprite_up_ram(11)(9)<="001";pacman_sprite_up_ram(11)(10)<="001";pacman_
sprite_up_ram(11)(11)<="001";pacman_sprite_up_ram(11)(12)<="001";pacman_sprite_
up_ram(11)(13)<="001";pacman_sprite_up_ram(11)(14)<="011";pacman_sprite_up_ram
(11)(15)<="011";

pacman_sprite_up_ram(12)(0)<="110";pacman_sprite_up_ram(12)(1)<="011";pacman_s
prite_up_ram(12)(2)<="011";pacman_sprite_up_ram(12)(3)<="001";pacman_sprite_up_
ram(12)(4)<="001";pacman_sprite_up_ram(12)(5)<="001";pacman_sprite_up_ram(12)(6
)<="001";pacman_sprite_up_ram(12)(7)<="001";pacman_sprite_up_ram(12)(8)<="001";
pacman_sprite_up_ram(12)(9)<="001";pacman_sprite_up_ram(12)(10)<="001";pacman_
sprite_up_ram(12)(11)<="001";pacman_sprite_up_ram(12)(12)<="001";pacman_sprite_
up_ram(12)(13)<="011";pacman_sprite_up_ram(12)(14)<="011";pacman_sprite_up_ram
(12)(15)<="011";

pacman_sprite_up_ram(13)(0)<="110";pacman_sprite_up_ram(13)(1)<="011";pacman_s
prite_up_ram(13)(2)<="011";pacman_sprite_up_ram(13)(3)<="011";pacman_sprite_up_
ram(13)(4)<="001";pacman_sprite_up_ram(13)(5)<="001";pacman_sprite_up_ram(13)(6
)<="001";pacman_sprite_up_ram(13)(7)<="001";pacman_sprite_up_ram(13)(8)<="001";
pacman_sprite_up_ram(13)(9)<="001";pacman_sprite_up_ram(13)(10)<="001";pacman_
sprite_up_ram(13)(11)<="001";pacman_sprite_up_ram(13)(12)<="011";pacman_sprite_
up_ram(13)(13)<="011";pacman_sprite_up_ram(13)(14)<="011";pacman_sprite_up_ram
(13)(15)<="011";

pacman_sprite_up_ram(14)(0)<="110";pacman_sprite_up_ram(14)(1)<="011";pacman_s
prite_up_ram(14)(2)<="011";pacman_sprite_up_ram(14)(3)<="011";pacman_sprite_up_
ram(14)(4)<="011";pacman_sprite_up_ram(14)(5)<="011";pacman_sprite_up_ram(14)(6
)<="001";pacman_sprite_up_ram(14)(7)<="001";pacman_sprite_up_ram(14)(8)<="001";
pacman_sprite_up_ram(14)(9)<="001";pacman_sprite_up_ram(14)(10)<="011";pacman_
sprite_up_ram(14)(11)<="011";pacman_sprite_up_ram(14)(12)<="011";pacman_sprite_
up_ram(14)(13)<="011";pacman_sprite_up_ram(14)(14)<="011";pacman_sprite_up_ram
(14)(15)<="011";

pacman_sprite_up_ram(15)(0)<="110";pacman_sprite_up_ram(15)(1)<="011";pacman_s
prite_up_ram(15)(2)<="011";pacman_sprite_up_ram(15)(3)<="011";pacman_sprite_up_
ram(15)(4)<="011";pacman_sprite_up_ram(15)(5)<="011";pacman_sprite_up_ram(15)(6
)<="011";pacman_sprite_up_ram(15)(7)<="011";pacman_sprite_up_ram(15)(8)<="011";
pacman_sprite_up_ram(15)(9)<="011";pacman_sprite_up_ram(15)(10)<="011";pacman_
sprite_up_ram(15)(11)<="011";pacman_sprite_up_ram(15)(12)<="011";pacman_sprite_
up_ram(15)(13)<="011";pacman_sprite_up_ram(15)(14)<="011";pacman_sprite_up_ram
(15)(15)<="011";


pacman_sprite_left_ram(0)(0)<="110";pacman_sprite_left_ram(0)(1)<="110";pacman_s
prite_left_ram(0)(2)<="110";pacman_sprite_left_ram(0)(3)<="110";pacman_sprite_left_
ram(0)(4)<="110";pacman_sprite_left_ram(0)(5)<="110";pacman_sprite_left_ram(0)(6)<
="110";pacman_sprite_left_ram(0)(7)<="110";pacman_sprite_left_ram(0)(8)<="110";pa
cman_sprite_left_ram(0)(9)<="110";pacman_sprite_left_ram(0)(10)<="110";pacman_spr
ite_left_ram(0)(11)<="110";pacman_sprite_left_ram(0)(12)<="110";pacman_sprite_left_
ram(0)(13)<="110";pacman_sprite_left_ram(0)(14)<="110";pacman_sprite_left_ram(0)(
15)<="110";

pacman_sprite_left_ram(1)(0)<="110";pacman_sprite_left_ram(1)(1)<="011";pacman_s
prite_left_ram(1)(2)<="011";pacman_sprite_left_ram(1)(3)<="011";pacman_sprite_left_
ram(1)(4)<="011";pacman_sprite_left_ram(1)(5)<="011";pacman_sprite_left_ram(1)(6)<
="001";pacman_sprite_left_ram(1)(7)<="001";pacman_sprite_left_ram(1)(8)<="001";pa
cman_sprite_left_ram(1)(9)<="001";pacman_sprite_left_ram(1)(10)<="011";pacman_spr
ite_left_ram(1)(11)<="011";pacman_sprite_left_ram(1)(12)<="011";pacman_sprite_left_
ram(1)(13)<="011";pacman_sprite_left_ram(1)(14)<="011";pacman_sprite_left_ram(1)(
15)<="011";

pacman_sprite_left_ram(2)(0)<="110";pacman_sprite_left_ram(2)(1)<="011";pacman_s
prite_left_ram(2)(2)<="011";pacman_sprite_left_ram(2)(3)<="011";pacman_sprite_left_
ram(2)(4)<="001";pacman_sprite_left_ram(2)(5)<="001";pacman_sprite_left_ram(2)(6)<
="001";pacman_sprite_left_ram(2)(7)<="001";pacman_sprite_left_ram(2)(8)<="001";pa
cman_sprite_left_ram(2)(9)<="001";pacman_sprite_left_ram(2)(10)<="001";pacman_spr
ite_left_ram(2)(11)<="001";pacman_sprite_left_ram(2)(12)<="011";pacman_sprite_left_
```

```
ram(2)(13)<="011";pacman_sprite_left_ram(2)(14)<="011";pacman_sprite_left_ram(2)(
15)<="011";

pacman_sprite_left_ram(3)(0)<="110";pacman_sprite_left_ram(3)(1)<="011";pacman_s
prite_left_ram(3)(2)<="011";pacman_sprite_left_ram(3)(3)<="001";pacman_sprite_left_
ram(3)(4)<="001";pacman_sprite_left_ram(3)(5)<="001";pacman_sprite_left_ram(3)(6)<
="001";pacman_sprite_left_ram(3)(7)<="001";pacman_sprite_left_ram(3)(8)<="001";pa
cman_sprite_left_ram(3)(9)<="001";pacman_sprite_left_ram(3)(10)<="001";pacman_spr
ite_left_ram(3)(11)<="001";pacman_sprite_left_ram(3)(12)<="001";pacman_sprite_left_
ram(3)(13)<="011";pacman_sprite_left_ram(3)(14)<="011";pacman_sprite_left_ram(3)(
15)<="011";

pacman_sprite_left_ram(4)(0)<="110";pacman_sprite_left_ram(4)(1)<="011";pacman_s
prite_left_ram(4)(2)<="011";pacman_sprite_left_ram(4)(3)<="011";pacman_sprite_left_
ram(4)(4)<="001";pacman_sprite_left_ram(4)(5)<="001";pacman_sprite_left_ram(4)(6)<
="001";pacman_sprite_left_ram(4)(7)<="001";pacman_sprite_left_ram(4)(8)<="001";pa
cman_sprite_left_ram(4)(9)<="001";pacman_sprite_left_ram(4)(10)<="001";pacman_spr
ite_left_ram(4)(11)<="001";pacman_sprite_left_ram(4)(12)<="001";pacman_sprite_left_
ram(4)(13)<="001";pacman_sprite_left_ram(4)(14)<="011";pacman_sprite_left_ram(4)(
15)<="011";

pacman_sprite_left_ram(5)(0)<="110";pacman_sprite_left_ram(5)(1)<="011";pacman_s
prite_left_ram(5)(2)<="011";pacman_sprite_left_ram(5)(3)<="011";pacman_sprite_left_
ram(5)(4)<="011";pacman_sprite_left_ram(5)(5)<="011";pacman_sprite_left_ram(5)(6)<
="001";pacman_sprite_left_ram(5)(7)<="001";pacman_sprite_left_ram(5)(8)<="001";pa
cman_sprite_left_ram(5)(9)<="001";pacman_sprite_left_ram(5)(10)<="001";pacman_spr
ite_left_ram(5)(11)<="001";pacman_sprite_left_ram(5)(12)<="001";pacman_sprite_left_
ram(5)(13)<="001";pacman_sprite_left_ram(5)(14)<="001";pacman_sprite_left_ram(5)(
15)<="011";

pacman_sprite_left_ram(6)(0)<="110";pacman_sprite_left_ram(6)(1)<="011";pacman_s
prite_left_ram(6)(2)<="011";pacman_sprite_left_ram(6)(3)<="011";pacman_sprite_left_
ram(6)(4)<="011";pacman_sprite_left_ram(6)(5)<="011";pacman_sprite_left_ram(6)(6)<
="011";pacman_sprite_left_ram(6)(7)<="011";pacman_sprite_left_ram(6)(8)<="001";pa
cman_sprite_left_ram(6)(9)<="001";pacman_sprite_left_ram(6)(10)<="001";pacman_spr
ite_left_ram(6)(11)<="001";pacman_sprite_left_ram(6)(12)<="001";pacman_sprite_left_
ram(6)(13)<="001";pacman_sprite_left_ram(6)(14)<="001";pacman_sprite_left_ram(6)(
15)<="011";

pacman_sprite_left_ram(7)(0)<="110";pacman_sprite_left_ram(7)(1)<="011";pacman_s
```

```
prite_left_ram(7)(2)<="011";pacman_sprite_left_ram(7)(3)<="011";pacman_sprite_left_
ram(7)(4)<="011";pacman_sprite_left_ram(7)(5)<="011";pacman_sprite_left_ram(7)(6)<
="011";pacman_sprite_left_ram(7)(7)<="011";pacman_sprite_left_ram(7)(8)<="001";pa
cman_sprite_left_ram(7)(9)<="001";pacman_sprite_left_ram(7)(10)<="001";pacman_spr
ite_left_ram(7)(11)<="001";pacman_sprite_left_ram(7)(12)<="001";pacman_sprite_left_
ram(7)(13)<="001";pacman_sprite_left_ram(7)(14)<="001";pacman_sprite_left_ram(7)(
15)<="011";

pacman_sprite_left_ram(8)(0)<="110";pacman_sprite_left_ram(8)(1)<="011";pacman_s
prite_left_ram(8)(2)<="011";pacman_sprite_left_ram(8)(3)<="011";pacman_sprite_left_
ram(8)(4)<="011";pacman_sprite_left_ram(8)(5)<="011";pacman_sprite_left_ram(8)(6)<
="011";pacman_sprite_left_ram(8)(7)<="001";pacman_sprite_left_ram(8)(8)<="001";pa
cman_sprite_left_ram(8)(9)<="001";pacman_sprite_left_ram(8)(10)<="001";pacman_spr
ite_left_ram(8)(11)<="001";pacman_sprite_left_ram(8)(12)<="001";pacman_sprite_left_
ram(8)(13)<="001";pacman_sprite_left_ram(8)(14)<="001";pacman_sprite_left_ram(8)(
15)<="011";

pacman_sprite_left_ram(9)(0)<="110";pacman_sprite_left_ram(9)(1)<="011";pacman_s
prite_left_ram(9)(2)<="011";pacman_sprite_left_ram(9)(3)<="011";pacman_sprite_left_
ram(9)(4)<="011";pacman_sprite_left_ram(9)(5)<="001";pacman_sprite_left_ram(9)(6)<
="001";pacman_sprite_left_ram(9)(7)<="001";pacman_sprite_left_ram(9)(8)<="001";pa
cman_sprite_left_ram(9)(9)<="001";pacman_sprite_left_ram(9)(10)<="001";pacman_spr
ite_left_ram(9)(11)<="001";pacman_sprite_left_ram(9)(12)<="001";pacman_sprite_left_
ram(9)(13)<="001";pacman_sprite_left_ram(9)(14)<="001";pacman_sprite_left_ram(9)(
15)<="011";

pacman_sprite_left_ram(10)(0)<="110";pacman_sprite_left_ram(10)(1)<="011";pacman
_sprite_left_ram(10)(2)<="011";pacman_sprite_left_ram(10)(3)<="001";pacman_sprite_
left_ram(10)(4)<="001";pacman_sprite_left_ram(10)(5)<="001";pacman_sprite_left_ram
(10)(6)<="001";pacman_sprite_left_ram(10)(7)<="001";pacman_sprite_left_ram(10)(8)<
="001";pacman_sprite_left_ram(10)(9)<="001";pacman_sprite_left_ram(10)(10)<="001"
;pacman_sprite_left_ram(10)(11)<="001";pacman_sprite_left_ram(10)(12)<="001";pacm
an_sprite_left_ram(10)(13)<="001";pacman_sprite_left_ram(10)(14)<="001";pacman_sp
rite_left_ram(10)(15)<="011";

pacman_sprite_left_ram(11)(0)<="110";pacman_sprite_left_ram(11)(1)<="011";pacman
_sprite_left_ram(11)(2)<="001";pacman_sprite_left_ram(11)(3)<="001";pacman_sprite_
left_ram(11)(4)<="001";pacman_sprite_left_ram(11)(5)<="001";pacman_sprite_left_ram
(11)(6)<="001";pacman_sprite_left_ram(11)(7)<="001";pacman_sprite_left_ram(11)(8)<
="001";pacman_sprite_left_ram(11)(9)<="001";pacman_sprite_left_ram(11)(10)<="001"
```

```
;pacman_sprite_left_ram(11)(11)<="001";pacman_sprite_left_ram(11)(12)<="001";pacm
an_sprite_left_ram(11)(13)<="001";pacman_sprite_left_ram(11)(14)<="011";pacman_sp
rite_left_ram(11)(15)<="011";

pacman_sprite_left_ram(12)(0)<="110";pacman_sprite_left_ram(12)(1)<="011";pacman
_sprite_left_ram(12)(2)<="011";pacman_sprite_left_ram(12)(3)<="001";pacman_sprite_
left_ram(12)(4)<="001";pacman_sprite_left_ram(12)(5)<="001";pacman_sprite_left_ram
(12)(6)<="001";pacman_sprite_left_ram(12)(7)<="001";pacman_sprite_left_ram(12)(8)<
="001";pacman_sprite_left_ram(12)(9)<="001";pacman_sprite_left_ram(12)(10)<="001"
;pacman_sprite_left_ram(12)(11)<="001";pacman_sprite_left_ram(12)(12)<="001";pacm
an_sprite_left_ram(12)(13)<="011";pacman_sprite_left_ram(12)(14)<="011";pacman_sp
rite_left_ram(12)(15)<="011";

pacman_sprite_left_ram(13)(0)<="110";pacman_sprite_left_ram(13)(1)<="011";pacman
_sprite_left_ram(13)(2)<="011";pacman_sprite_left_ram(13)(3)<="011";pacman_sprite_
left_ram(13)(4)<="001";pacman_sprite_left_ram(13)(5)<="001";pacman_sprite_left_ram
(13)(6)<="001";pacman_sprite_left_ram(13)(7)<="001";pacman_sprite_left_ram(13)(8)<
="001";pacman_sprite_left_ram(13)(9)<="001";pacman_sprite_left_ram(13)(10)<="001"
;pacman_sprite_left_ram(13)(11)<="001";pacman_sprite_left_ram(13)(12)<="011";pacm
an_sprite_left_ram(13)(13)<="011";pacman_sprite_left_ram(13)(14)<="011";pacman_sp
rite_left_ram(13)(15)<="011";

pacman_sprite_left_ram(14)(0)<="110";pacman_sprite_left_ram(14)(1)<="011";pacman
_sprite_left_ram(14)(2)<="011";pacman_sprite_left_ram(14)(3)<="011";pacman_sprite_
left_ram(14)(4)<="011";pacman_sprite_left_ram(14)(5)<="011";pacman_sprite_left_ram
(14)(6)<="001";pacman_sprite_left_ram(14)(7)<="001";pacman_sprite_left_ram(14)(8)<
="001";pacman_sprite_left_ram(14)(9)<="001";pacman_sprite_left_ram(14)(10)<="011"
;pacman_sprite_left_ram(14)(11)<="011";pacman_sprite_left_ram(14)(12)<="011";pacm
an_sprite_left_ram(14)(13)<="011";pacman_sprite_left_ram(14)(14)<="011";pacman_sp
rite_left_ram(14)(15)<="011";

pacman_sprite_left_ram(15)(0)<="110";pacman_sprite_left_ram(15)(1)<="011";pacman
_sprite_left_ram(15)(2)<="011";pacman_sprite_left_ram(15)(3)<="011";pacman_sprite_
left_ram(15)(4)<="011";pacman_sprite_left_ram(15)(5)<="011";pacman_sprite_left_ram
(15)(6)<="011";pacman_sprite_left_ram(15)(7)<="011";pacman_sprite_left_ram(15)(8)<
="011";pacman_sprite_left_ram(15)(9)<="011";pacman_sprite_left_ram(15)(10)<="011"
;pacman_sprite_left_ram(15)(11)<="011";pacman_sprite_left_ram(15)(12)<="011";pacm
an_sprite_left_ram(15)(13)<="011";pacman_sprite_left_ram(15)(14)<="011";pacman_sp
rite_left_ram(15)(15)<="011";
```

```
ghost_sprite_1_ram(0)(0)<="000";ghost_sprite_1_ram(0)(1)<="000";ghost_sprite_1_ram
(0)(2)<="000";ghost_sprite_1_ram(0)(3)<="000";ghost_sprite_1_ram(0)(4)<="000";ghos
t_sprite_1_ram(0)(5)<="100";ghost_sprite_1_ram(0)(6)<="100";ghost_sprite_1_ram(0)(
7)<="100";ghost_sprite_1_ram(0)(8)<="100";ghost_sprite_1_ram(0)(9)<="100";ghost_s
prite_1_ram(0)(10)<="100";ghost_sprite_1_ram(0)(11)<="000";ghost_sprite_1_ram(0)(1
2)<="000";ghost_sprite_1_ram(0)(13)<="000";ghost_sprite_1_ram(0)(14)<="000";ghost
_sprite_1_ram(0)(15)<="000";

ghost_sprite_1_ram(1)(0)<="000";ghost_sprite_1_ram(1)(1)<="000";ghost_sprite_1_ram
(1)(2)<="000";ghost_sprite_1_ram(1)(3)<="000";ghost_sprite_1_ram(1)(4)<="100";ghos
t_sprite_1_ram(1)(5)<="100";ghost_sprite_1_ram(1)(6)<="100";ghost_sprite_1_ram(1)(
7)<="100";ghost_sprite_1_ram(1)(8)<="100";ghost_sprite_1_ram(1)(9)<="100";ghost_s
prite_1_ram(1)(10)<="100";ghost_sprite_1_ram(1)(11)<="100";ghost_sprite_1_ram(1)(1
2)<="000";ghost_sprite_1_ram(1)(13)<="000";ghost_sprite_1_ram(1)(14)<="000";ghost
_sprite_1_ram(1)(15)<="000";

ghost_sprite_1_ram(2)(0)<="000";ghost_sprite_1_ram(2)(1)<="000";ghost_sprite_1_ram
(2)(2)<="000";ghost_sprite_1_ram(2)(3)<="100";ghost_sprite_1_ram(2)(4)<="100";ghos
t_sprite_1_ram(2)(5)<="100";ghost_sprite_1_ram(2)(6)<="100";ghost_sprite_1_ram(2)(
7)<="100";ghost_sprite_1_ram(2)(8)<="100";ghost_sprite_1_ram(2)(9)<="100";ghost_s
prite_1_ram(2)(10)<="100";ghost_sprite_1_ram(2)(11)<="100";ghost_sprite_1_ram(2)(1
2)<="100";ghost_sprite_1_ram(2)(13)<="000";ghost_sprite_1_ram(2)(14)<="000";ghost
_sprite_1_ram(2)(15)<="000";

ghost_sprite_1_ram(3)(0)<="000";ghost_sprite_1_ram(3)(1)<="000";ghost_sprite_1_ram
(3)(2)<="100";ghost_sprite_1_ram(3)(3)<="100";ghost_sprite_1_ram(3)(4)<="100";ghos
t_sprite_1_ram(3)(5)<="100";ghost_sprite_1_ram(3)(6)<="100";ghost_sprite_1_ram(3)(
7)<="100";ghost_sprite_1_ram(3)(8)<="100";ghost_sprite_1_ram(3)(9)<="100";ghost_s
prite_1_ram(3)(10)<="100";ghost_sprite_1_ram(3)(11)<="100";ghost_sprite_1_ram(3)(1
2)<="100";ghost_sprite_1_ram(3)(13)<="100";ghost_sprite_1_ram(3)(14)<="000";ghost
_sprite_1_ram(3)(15)<="000";

ghost_sprite_1_ram(4)(0)<="000";ghost_sprite_1_ram(4)(1)<="100";ghost_sprite_1_ram
(4)(2)<="110";ghost_sprite_1_ram(4)(3)<="110";ghost_sprite_1_ram(4)(4)<="100";ghos
t_sprite_1_ram(4)(5)<="100";ghost_sprite_1_ram(4)(6)<="100";ghost_sprite_1_ram(4)(
7)<="100";ghost_sprite_1_ram(4)(8)<="100";ghost_sprite_1_ram(4)(9)<="110";ghost_s
prite_1_ram(4)(10)<="110";ghost_sprite_1_ram(4)(11)<="110";ghost_sprite_1_ram(4)(1
2)<="100";ghost_sprite_1_ram(4)(13)<="100";ghost_sprite_1_ram(4)(14)<="100";ghost
_sprite_1_ram(4)(15)<="000";
```

```
ghost_sprite_1_ram(5)(0)<="110";ghost_sprite_1_ram(5)(1)<="110";ghost_sprite_1_ram
(5)(2)<="110";ghost_sprite_1_ram(5)(3)<="110";ghost_sprite_1_ram(5)(4)<="110";ghos
t_sprite_1_ram(5)(5)<="100";ghost_sprite_1_ram(5)(6)<="100";ghost_sprite_1_ram(5)(
7)<="100";ghost_sprite_1_ram(5)(8)<="110";ghost_sprite_1_ram(5)(9)<="110";ghost_s
prite_1_ram(5)(10)<="110";ghost_sprite_1_ram(5)(11)<="110";ghost_sprite_1_ram(5)(1
2)<="110";ghost_sprite_1_ram(5)(13)<="100";ghost_sprite_1_ram(5)(14)<="100";ghost
_sprite_1_ram(5)(15)<="000";

ghost_sprite_1_ram(6)(0)<="110";ghost_sprite_1_ram(6)(1)<="110";ghost_sprite_1_ram
(6)(2)<="000";ghost_sprite_1_ram(6)(3)<="110";ghost_sprite_1_ram(6)(4)<="110";ghos
t_sprite_1_ram(6)(5)<="100";ghost_sprite_1_ram(6)(6)<="100";ghost_sprite_1_ram(6)(
7)<="100";ghost_sprite_1_ram(6)(8)<="110";ghost_sprite_1_ram(6)(9)<="110";ghost_s
prite_1_ram(6)(10)<="000";ghost_sprite_1_ram(6)(11)<="110";ghost_sprite_1_ram(6)(1
2)<="110";ghost_sprite_1_ram(6)(13)<="100";ghost_sprite_1_ram(6)(14)<="100";ghost
_sprite_1_ram(6)(15)<="000";

ghost_sprite_1_ram(7)(0)<="110";ghost_sprite_1_ram(7)(1)<="000";ghost_sprite_1_ram
(7)(2)<="000";ghost_sprite_1_ram(7)(3)<="000";ghost_sprite_1_ram(7)(4)<="110";ghos
t_sprite_1_ram(7)(5)<="100";ghost_sprite_1_ram(7)(6)<="100";ghost_sprite_1_ram(7)(
7)<="100";ghost_sprite_1_ram(7)(8)<="110";ghost_sprite_1_ram(7)(9)<="000";ghost_s
prite_1_ram(7)(10)<="000";ghost_sprite_1_ram(7)(11)<="000";ghost_sprite_1_ram(7)(1
2)<="110";ghost_sprite_1_ram(7)(13)<="100";ghost_sprite_1_ram(7)(14)<="100";ghost
_sprite_1_ram(7)(15)<="000";

ghost_sprite_1_ram(8)(0)<="000";ghost_sprite_1_ram(8)(1)<="000";ghost_sprite_1_ram
(8)(2)<="000";ghost_sprite_1_ram(8)(3)<="000";ghost_sprite_1_ram(8)(4)<="100";ghos
t_sprite_1_ram(8)(5)<="100";ghost_sprite_1_ram(8)(6)<="100";ghost_sprite_1_ram(8)(
7)<="100";ghost_sprite_1_ram(8)(8)<="100";ghost_sprite_1_ram(8)(9)<="000";ghost_s
prite_1_ram(8)(10)<="000";ghost_sprite_1_ram(8)(11)<="000";ghost_sprite_1_ram(8)(1
2)<="100";ghost_sprite_1_ram(8)(13)<="100";ghost_sprite_1_ram(8)(14)<="100";ghost
_sprite_1_ram(8)(15)<="000";

ghost_sprite_1_ram(9)(0)<="000";ghost_sprite_1_ram(9)(1)<="100";ghost_sprite_1_ram
(9)(2)<="000";ghost_sprite_1_ram(9)(3)<="100";ghost_sprite_1_ram(9)(4)<="100";ghos
t_sprite_1_ram(9)(5)<="100";ghost_sprite_1_ram(9)(6)<="100";ghost_sprite_1_ram(9)(
7)<="100";ghost_sprite_1_ram(9)(8)<="100";ghost_sprite_1_ram(9)(9)<="100";ghost_s
prite_1_ram(9)(10)<="000";ghost_sprite_1_ram(9)(11)<="100";ghost_sprite_1_ram(9)(1
2)<="100";ghost_sprite_1_ram(9)(13)<="100";ghost_sprite_1_ram(9)(14)<="100";ghost
_sprite_1_ram(9)(15)<="000";
```

```
ghost_sprite_1_ram(10)(0)<="000";ghost_sprite_1_ram(10)(1)<="100";ghost_sprite_1_r
am(10)(2)<="100";ghost_sprite_1_ram(10)(3)<="100";ghost_sprite_1_ram(10)(4)<="10
0";ghost_sprite_1_ram(10)(5)<="100";ghost_sprite_1_ram(10)(6)<="100";ghost_sprite_
1_ram(10)(7)<="100";ghost_sprite_1_ram(10)(8)<="100";ghost_sprite_1_ram(10)(9)<="
100";ghost_sprite_1_ram(10)(10)<="100";ghost_sprite_1_ram(10)(11)<="100";ghost_sp
rite_1_ram(10)(12)<="100";ghost_sprite_1_ram(10)(13)<="100";ghost_sprite_1_ram(10
)(14)<="100";ghost_sprite_1_ram(10)(15)<="000";

ghost_sprite_1_ram(11)(0)<="000";ghost_sprite_1_ram(11)(1)<="100";ghost_sprite_1_r
am(11)(2)<="100";ghost_sprite_1_ram(11)(3)<="100";ghost_sprite_1_ram(11)(4)<="10
0";ghost_sprite_1_ram(11)(5)<="100";ghost_sprite_1_ram(11)(6)<="100";ghost_sprite_
1_ram(11)(7)<="100";ghost_sprite_1_ram(11)(8)<="100";ghost_sprite_1_ram(11)(9)<="
100";ghost_sprite_1_ram(11)(10)<="100";ghost_sprite_1_ram(11)(11)<="100";ghost_sp
rite_1_ram(11)(12)<="100";ghost_sprite_1_ram(11)(13)<="100";ghost_sprite_1_ram(11
)(14)<="100";ghost_sprite_1_ram(11)(15)<="000";

ghost_sprite_1_ram(12)(0)<="000";ghost_sprite_1_ram(12)(1)<="100";ghost_sprite_1_r
am(12)(2)<="100";ghost_sprite_1_ram(12)(3)<="100";ghost_sprite_1_ram(12)(4)<="10
0";ghost_sprite_1_ram(12)(5)<="100";ghost_sprite_1_ram(12)(6)<="100";ghost_sprite_
1_ram(12)(7)<="100";ghost_sprite_1_ram(12)(8)<="100";ghost_sprite_1_ram(12)(9)<="
100";ghost_sprite_1_ram(12)(10)<="100";ghost_sprite_1_ram(12)(11)<="100";ghost_sp
rite_1_ram(12)(12)<="100";ghost_sprite_1_ram(12)(13)<="100";ghost_sprite_1_ram(12
)(14)<="100";ghost_sprite_1_ram(12)(15)<="000";

ghost_sprite_1_ram(13)(0)<="000";ghost_sprite_1_ram(13)(1)<="100";ghost_sprite_1_r
am(13)(2)<="100";ghost_sprite_1_ram(13)(3)<="100";ghost_sprite_1_ram(13)(4)<="10
0";ghost_sprite_1_ram(13)(5)<="100";ghost_sprite_1_ram(13)(6)<="100";ghost_sprite_
1_ram(13)(7)<="100";ghost_sprite_1_ram(13)(8)<="100";ghost_sprite_1_ram(13)(9)<="
100";ghost_sprite_1_ram(13)(10)<="100";ghost_sprite_1_ram(13)(11)<="100";ghost_sp
rite_1_ram(13)(12)<="100";ghost_sprite_1_ram(13)(13)<="100";ghost_sprite_1_ram(13
)(14)<="100";ghost_sprite_1_ram(13)(15)<="000";

ghost_sprite_1_ram(14)(0)<="000";ghost_sprite_1_ram(14)(1)<="100";ghost_sprite_1_r
am(14)(2)<="100";ghost_sprite_1_ram(14)(3)<="100";ghost_sprite_1_ram(14)(4)<="00
0";ghost_sprite_1_ram(14)(5)<="100";ghost_sprite_1_ram(14)(6)<="100";ghost_sprite_
1_ram(14)(7)<="100";ghost_sprite_1_ram(14)(8)<="000";ghost_sprite_1_ram(14)(9)<="
100";ghost_sprite_1_ram(14)(10)<="100";ghost_sprite_1_ram(14)(11)<="100";ghost_sp
rite_1_ram(14)(12)<="000";ghost_sprite_1_ram(14)(13)<="100";ghost_sprite_1_ram(14
)(14)<="100";ghost_sprite_1_ram(14)(15)<="000";
```

```
ghost_sprite_1_ram(15)(0)<="100";ghost_sprite_1_ram(15)(1)<="100";ghost_sprite_1_r
am(15)(2)<="000";ghost_sprite_1_ram(15)(3)<="000";ghost_sprite_1_ram(15)(4)<="00
0";ghost_sprite_1_ram(15)(5)<="100";ghost_sprite_1_ram(15)(6)<="000";ghost_sprite_
1_ram(15)(7)<="000";ghost_sprite_1_ram(15)(8)<="000";ghost_sprite_1_ram(15)(9)<="
100";ghost_sprite_1_ram(15)(10)<="000";ghost_sprite_1_ram(15)(11)<="000";ghost_sp
rite_1_ram(15)(12)<="100";ghost_sprite_1_ram(15)(13)<="100";ghost_sprite_1_ram(15
)(14)<="000";ghost_sprite_1_ram(15)(15)<="000";


square_sprite_ram(0)(0)<="110";square_sprite_ram(0)(1)<="110";square_sprite_ram(0)(
2)<="110";square_sprite_ram(0)(3)<="110";square_sprite_ram(0)(4)<="110";square_spr
ite_ram(0)(5)<="110";square_sprite_ram(0)(6)<="110";square_sprite_ram(0)(7)<="110";
square_sprite_ram(0)(8)<="110";square_sprite_ram(0)(9)<="110";square_sprite_ram(0)(
10)<="110";square_sprite_ram(0)(11)<="110";square_sprite_ram(0)(12)<="110";square
_sprite_ram(0)(13)<="110";square_sprite_ram(0)(14)<="110";square_sprite_ram(0)(15)
<="110";

square_sprite_ram(1)(0)<="110";square_sprite_ram(1)(1)<="011";square_sprite_ram(1)(
2)<="011";square_sprite_ram(1)(3)<="011";square_sprite_ram(1)(4)<="011";square_spr
ite_ram(1)(5)<="011";square_sprite_ram(1)(6)<="011";square_sprite_ram(1)(7)<="011";
square_sprite_ram(1)(8)<="011";square_sprite_ram(1)(9)<="011";square_sprite_ram(1)(
10)<="011";square_sprite_ram(1)(11)<="011";square_sprite_ram(1)(12)<="011";square
_sprite_ram(1)(13)<="011";square_sprite_ram(1)(14)<="011";square_sprite_ram(1)(15)
<="011";

square_sprite_ram(2)(0)<="110";square_sprite_ram(2)(1)<="011";square_sprite_ram(2)(
2)<="011";square_sprite_ram(2)(3)<="011";square_sprite_ram(2)(4)<="011";square_spr
ite_ram(2)(5)<="011";square_sprite_ram(2)(6)<="011";square_sprite_ram(2)(7)<="011";
square_sprite_ram(2)(8)<="011";square_sprite_ram(2)(9)<="011";square_sprite_ram(2)(
10)<="011";square_sprite_ram(2)(11)<="011";square_sprite_ram(2)(12)<="011";square
_sprite_ram(2)(13)<="011";square_sprite_ram(2)(14)<="011";square_sprite_ram(2)(15)
<="011";

square_sprite_ram(3)(0)<="110";square_sprite_ram(3)(1)<="011";square_sprite_ram(3)(
2)<="011";square_sprite_ram(3)(3)<="011";square_sprite_ram(3)(4)<="011";square_spr
ite_ram(3)(5)<="011";square_sprite_ram(3)(6)<="011";square_sprite_ram(3)(7)<="011";
square_sprite_ram(3)(8)<="011";square_sprite_ram(3)(9)<="011";square_sprite_ram(3)(
10)<="011";square_sprite_ram(3)(11)<="011";square_sprite_ram(3)(12)<="011";square
_sprite_ram(3)(13)<="011";square_sprite_ram(3)(14)<="011";square_sprite_ram(3)(15)
```

```
<="011";

square_sprite_ram(4)(0)<="110";square_sprite_ram(4)(1)<="011";square_sprite_ram(4)(
2)<="011";square_sprite_ram(4)(3)<="011";square_sprite_ram(4)(4)<="011";square_spr
ite_ram(4)(5)<="011";square_sprite_ram(4)(6)<="011";square_sprite_ram(4)(7)<="011";
square_sprite_ram(4)(8)<="011";square_sprite_ram(4)(9)<="011";square_sprite_ram(4)(
10)<="011";square_sprite_ram(4)(11)<="011";square_sprite_ram(4)(12)<="011";square
_sprite_ram(4)(13)<="011";square_sprite_ram(4)(14)<="011";square_sprite_ram(4)(15)
<="011";

square_sprite_ram(5)(0)<="110";square_sprite_ram(5)(1)<="011";square_sprite_ram(5)(
2)<="011";square_sprite_ram(5)(3)<="011";square_sprite_ram(5)(4)<="011";square_spr
ite_ram(5)(5)<="011";square_sprite_ram(5)(6)<="011";square_sprite_ram(5)(7)<="011";
square_sprite_ram(5)(8)<="011";square_sprite_ram(5)(9)<="011";square_sprite_ram(5)(
10)<="011";square_sprite_ram(5)(11)<="011";square_sprite_ram(5)(12)<="011";square
_sprite_ram(5)(13)<="011";square_sprite_ram(5)(14)<="011";square_sprite_ram(5)(15)
<="011";

square_sprite_ram(6)(0)<="110";square_sprite_ram(6)(1)<="011";square_sprite_ram(6)(
2)<="011";square_sprite_ram(6)(3)<="011";square_sprite_ram(6)(4)<="011";square_spr
ite_ram(6)(5)<="011";square_sprite_ram(6)(6)<="011";square_sprite_ram(6)(7)<="011";
square_sprite_ram(6)(8)<="011";square_sprite_ram(6)(9)<="011";square_sprite_ram(6)(
10)<="011";square_sprite_ram(6)(11)<="011";square_sprite_ram(6)(12)<="011";square
_sprite_ram(6)(13)<="011";square_sprite_ram(6)(14)<="011";square_sprite_ram(6)(15)
<="011";

square_sprite_ram(7)(0)<="110";square_sprite_ram(7)(1)<="011";square_sprite_ram(7)(
2)<="011";square_sprite_ram(7)(3)<="011";square_sprite_ram(7)(4)<="011";square_spr
ite_ram(7)(5)<="011";square_sprite_ram(7)(6)<="011";square_sprite_ram(7)(7)<="011";
square_sprite_ram(7)(8)<="011";square_sprite_ram(7)(9)<="011";square_sprite_ram(7)(
10)<="011";square_sprite_ram(7)(11)<="011";square_sprite_ram(7)(12)<="011";square
_sprite_ram(7)(13)<="011";square_sprite_ram(7)(14)<="011";square_sprite_ram(7)(15)
<="011";

square_sprite_ram(8)(0)<="110";square_sprite_ram(8)(1)<="011";square_sprite_ram(8)(
2)<="011";square_sprite_ram(8)(3)<="011";square_sprite_ram(8)(4)<="011";square_spr
ite_ram(8)(5)<="011";square_sprite_ram(8)(6)<="011";square_sprite_ram(8)(7)<="011";
square_sprite_ram(8)(8)<="011";square_sprite_ram(8)(9)<="011";square_sprite_ram(8)(
10)<="011";square_sprite_ram(8)(11)<="011";square_sprite_ram(8)(12)<="011";square
_sprite_ram(8)(13)<="011";square_sprite_ram(8)(14)<="011";square_sprite_ram(8)(15)
```

```
<="011";

square_sprite_ram(9)(0)<="110";square_sprite_ram(9)(1)<="011";square_sprite_ram(9)(
2)<="011";square_sprite_ram(9)(3)<="011";square_sprite_ram(9)(4)<="011";square_spr
ite_ram(9)(5)<="011";square_sprite_ram(9)(6)<="011";square_sprite_ram(9)(7)<="011";
square_sprite_ram(9)(8)<="011";square_sprite_ram(9)(9)<="011";square_sprite_ram(9)(
10)<="011";square_sprite_ram(9)(11)<="011";square_sprite_ram(9)(12)<="011";square
_sprite_ram(9)(13)<="011";square_sprite_ram(9)(14)<="011";square_sprite_ram(9)(15)
<="011";

square_sprite_ram(10)(0)<="110";square_sprite_ram(10)(1)<="011";square_sprite_ram(
10)(2)<="011";square_sprite_ram(10)(3)<="011";square_sprite_ram(10)(4)<="011";squa
re_sprite_ram(10)(5)<="011";square_sprite_ram(10)(6)<="011";square_sprite_ram(10)(7
)<="011";square_sprite_ram(10)(8)<="011";square_sprite_ram(10)(9)<="011";square_sp
rite_ram(10)(10)<="011";square_sprite_ram(10)(11)<="011";square_sprite_ram(10)(12)
<="011";square_sprite_ram(10)(13)<="011";square_sprite_ram(10)(14)<="011";square_
sprite_ram(10)(15)<="011";

square_sprite_ram(11)(0)<="110";square_sprite_ram(11)(1)<="011";square_sprite_ram(
11)(2)<="011";square_sprite_ram(11)(3)<="011";square_sprite_ram(11)(4)<="011";squa
re_sprite_ram(11)(5)<="011";square_sprite_ram(11)(6)<="011";square_sprite_ram(11)(7
)<="011";square_sprite_ram(11)(8)<="011";square_sprite_ram(11)(9)<="011";square_sp
rite_ram(11)(10)<="011";square_sprite_ram(11)(11)<="011";square_sprite_ram(11)(12)
<="011";square_sprite_ram(11)(13)<="011";square_sprite_ram(11)(14)<="011";square_
sprite_ram(11)(15)<="011";

square_sprite_ram(12)(0)<="110";square_sprite_ram(12)(1)<="011";square_sprite_ram(
12)(2)<="011";square_sprite_ram(12)(3)<="011";square_sprite_ram(12)(4)<="011";squa
re_sprite_ram(12)(5)<="011";square_sprite_ram(12)(6)<="011";square_sprite_ram(12)(7
)<="011";square_sprite_ram(12)(8)<="011";square_sprite_ram(12)(9)<="011";square_sp
rite_ram(12)(10)<="011";square_sprite_ram(12)(11)<="011";square_sprite_ram(12)(12)
<="011";square_sprite_ram(12)(13)<="011";square_sprite_ram(12)(14)<="011";square_
sprite_ram(12)(15)<="011";

square_sprite_ram(13)(0)<="110";square_sprite_ram(13)(1)<="011";square_sprite_ram(
13)(2)<="011";square_sprite_ram(13)(3)<="011";square_sprite_ram(13)(4)<="011";squa
re_sprite_ram(13)(5)<="011";square_sprite_ram(13)(6)<="011";square_sprite_ram(13)(7
)<="011";square_sprite_ram(13)(8)<="011";square_sprite_ram(13)(9)<="011";square_sp
rite_ram(13)(10)<="011";square_sprite_ram(13)(11)<="011";square_sprite_ram(13)(12)
<="011";square_sprite_ram(13)(13)<="011";square_sprite_ram(13)(14)<="011";square_
```

sprite_ram(13)(15)<="011";

square_sprite_ram(14)(0)<="110";square_sprite_ram(14)(1)<="011";square_sprite_ram(14)(2)<="011";square_sprite_ram(14)(3)<="011";square_sprite_ram(14)(4)<="011";square_sprite_ram(14)(5)<="011";square_sprite_ram(14)(6)<="011";square_sprite_ram(14)(7)<="011";square_sprite_ram(14)(8)<="011";square_sprite_ram(14)(9)<="011";square_sprite_ram(14)(10)<="011";square_sprite_ram(14)(11)<="011";square_sprite_ram(14)(12)<="011";square_sprite_ram(14)(13)<="011";square_sprite_ram(14)(14)<="011";square_sprite_ram(14)(15)<="011";

square_sprite_ram(15)(0)<="110";square_sprite_ram(15)(1)<="011";square_sprite_ram(15)(2)<="011";square_sprite_ram(15)(3)<="011";square_sprite_ram(15)(4)<="011";square_sprite_ram(15)(5)<="011";square_sprite_ram(15)(6)<="011";square_sprite_ram(15)(7)<="011";square_sprite_ram(15)(8)<="011";square_sprite_ram(15)(9)<="011";square_sprite_ram(15)(10)<="011";square_sprite_ram(15)(11)<="011";square_sprite_ram(15)(12)<="011";square_sprite_ram(15)(13)<="011";square_sprite_ram(15)(14)<="011";square_sprite_ram(15)(15)<="011";

black_sprite_ram(0)(0)<="000";black_sprite_ram(0)(1)<="000";black_sprite_ram(0)(2)<="000";black_sprite_ram(0)(3)<="000";black_sprite_ram(0)(4)<="000";black_sprite_ram(0)(5)<="000";black_sprite_ram(0)(6)<="000";black_sprite_ram(0)(7)<="000";black_sprite_ram(0)(8)<="000";black_sprite_ram(0)(9)<="000";black_sprite_ram(0)(10)<="000";black_sprite_ram(0)(11)<="000";black_sprite_ram(0)(12)<="000";black_sprite_ram(0)(13)<="000";black_sprite_ram(0)(14)<="000";black_sprite_ram(0)(15)<="000";

black_sprite_ram(1)(0)<="000";black_sprite_ram(1)(1)<="000";black_sprite_ram(1)(2)<="000";black_sprite_ram(1)(3)<="000";black_sprite_ram(1)(4)<="000";black_sprite_ram(1)(5)<="000";black_sprite_ram(1)(6)<="000";black_sprite_ram(1)(7)<="000";black_sprite_ram(1)(8)<="000";black_sprite_ram(1)(9)<="000";black_sprite_ram(1)(10)<="000";black_sprite_ram(1)(11)<="000";black_sprite_ram(1)(12)<="000";black_sprite_ram(1)(13)<="000";black_sprite_ram(1)(14)<="000";black_sprite_ram(1)(15)<="000";

black_sprite_ram(2)(0)<="000";black_sprite_ram(2)(1)<="000";black_sprite_ram(2)(2)<="000";black_sprite_ram(2)(3)<="000";black_sprite_ram(2)(4)<="000";black_sprite_ram(2)(5)<="000";black_sprite_ram(2)(6)<="000";black_sprite_ram(2)(7)<="000";black_sprite_ram(2)(8)<="000";black_sprite_ram(2)(9)<="000";black_sprite_ram(2)(10)<="000";black_sprite_ram(2)(11)<="000";black_sprite_ram(2)(12)<="000";black_sprite_ram(2)(13)<="000";black_sprite_ram(2)(14)<="000";black_sprite_ram(2)(15)<="000";

```
black_sprite_ram(3)(0)<="000";black_sprite_ram(3)(1)<="000";black_sprite_ram(3)(2)<
="000";black_sprite_ram(3)(3)<="000";black_sprite_ram(3)(4)<="000";black_sprite_ra
m(3)(5)<="000";black_sprite_ram(3)(6)<="000";black_sprite_ram(3)(7)<="000";black_s
prite_ram(3)(8)<="000";black_sprite_ram(3)(9)<="000";black_sprite_ram(3)(10)<="000
";black_sprite_ram(3)(11)<="000";black_sprite_ram(3)(12)<="000";black_sprite_ram(3)
(13)<="000";black_sprite_ram(3)(14)<="000";black_sprite_ram(3)(15)<="000";

black_sprite_ram(4)(0)<="000";black_sprite_ram(4)(1)<="000";black_sprite_ram(4)(2)<
="000";black_sprite_ram(4)(3)<="000";black_sprite_ram(4)(4)<="000";black_sprite_ra
m(4)(5)<="000";black_sprite_ram(4)(6)<="000";black_sprite_ram(4)(7)<="000";black_s
prite_ram(4)(8)<="000";black_sprite_ram(4)(9)<="000";black_sprite_ram(4)(10)<="000
";black_sprite_ram(4)(11)<="000";black_sprite_ram(4)(12)<="000";black_sprite_ram(4)
(13)<="000";black_sprite_ram(4)(14)<="000";black_sprite_ram(4)(15)<="000";

black_sprite_ram(5)(0)<="000";black_sprite_ram(5)(1)<="000";black_sprite_ram(5)(2)<
="000";black_sprite_ram(5)(3)<="000";black_sprite_ram(5)(4)<="000";black_sprite_ra
m(5)(5)<="000";black_sprite_ram(5)(6)<="000";black_sprite_ram(5)(7)<="000";black_s
prite_ram(5)(8)<="000";black_sprite_ram(5)(9)<="000";black_sprite_ram(5)(10)<="000
";black_sprite_ram(5)(11)<="000";black_sprite_ram(5)(12)<="000";black_sprite_ram(5)
(13)<="000";black_sprite_ram(5)(14)<="000";black_sprite_ram(5)(15)<="000";

black_sprite_ram(6)(0)<="000";black_sprite_ram(6)(1)<="000";black_sprite_ram(6)(2)<
="000";black_sprite_ram(6)(3)<="000";black_sprite_ram(6)(4)<="000";black_sprite_ra
m(6)(5)<="000";black_sprite_ram(6)(6)<="000";black_sprite_ram(6)(7)<="000";black_s
prite_ram(6)(8)<="000";black_sprite_ram(6)(9)<="000";black_sprite_ram(6)(10)<="000
";black_sprite_ram(6)(11)<="000";black_sprite_ram(6)(12)<="000";black_sprite_ram(6)
(13)<="000";black_sprite_ram(6)(14)<="000";black_sprite_ram(6)(15)<="000";

black_sprite_ram(7)(0)<="000";black_sprite_ram(7)(1)<="000";black_sprite_ram(7)(2)<
="000";black_sprite_ram(7)(3)<="000";black_sprite_ram(7)(4)<="000";black_sprite_ra
m(7)(5)<="000";black_sprite_ram(7)(6)<="000";black_sprite_ram(7)(7)<="000";black_s
prite_ram(7)(8)<="000";black_sprite_ram(7)(9)<="000";black_sprite_ram(7)(10)<="000
";black_sprite_ram(7)(11)<="000";black_sprite_ram(7)(12)<="000";black_sprite_ram(7)
(13)<="000";black_sprite_ram(7)(14)<="000";black_sprite_ram(7)(15)<="000";

black_sprite_ram(8)(0)<="000";black_sprite_ram(8)(1)<="000";black_sprite_ram(8)(2)<
="000";black_sprite_ram(8)(3)<="000";black_sprite_ram(8)(4)<="000";black_sprite_ra
m(8)(5)<="000";black_sprite_ram(8)(6)<="000";black_sprite_ram(8)(7)<="000";black_s
prite_ram(8)(8)<="000";black_sprite_ram(8)(9)<="000";black_sprite_ram(8)(10)<="000
";black_sprite_ram(8)(11)<="000";black_sprite_ram(8)(12)<="000";black_sprite_ram(8)
```

```
(13)<="000";black_sprite_ram(8)(14)<="000";black_sprite_ram(8)(15)<="000";

black_sprite_ram(9)(0)<="000";black_sprite_ram(9)(1)<="000";black_sprite_ram(9)(2)<
="000";black_sprite_ram(9)(3)<="000";black_sprite_ram(9)(4)<="000";black_sprite_ra
m(9)(5)<="000";black_sprite_ram(9)(6)<="000";black_sprite_ram(9)(7)<="000";black_s
prite_ram(9)(8)<="000";black_sprite_ram(9)(9)<="000";black_sprite_ram(9)(10)<="000
";black_sprite_ram(9)(11)<="000";black_sprite_ram(9)(12)<="000";black_sprite_ram(9)
(13)<="000";black_sprite_ram(9)(14)<="000";black_sprite_ram(9)(15)<="000";

black_sprite_ram(10)(0)<="000";black_sprite_ram(10)(1)<="000";black_sprite_ram(10)(
2)<="000";black_sprite_ram(10)(3)<="000";black_sprite_ram(10)(4)<="000";black_sprit
e_ram(10)(5)<="000";black_sprite_ram(10)(6)<="000";black_sprite_ram(10)(7)<="000"
;black_sprite_ram(10)(8)<="000";black_sprite_ram(10)(9)<="000";black_sprite_ram(10)
(10)<="000";black_sprite_ram(10)(11)<="000";black_sprite_ram(10)(12)<="000";black
_sprite_ram(10)(13)<="000";black_sprite_ram(10)(14)<="000";black_sprite_ram(10)(15
)<="000";

black_sprite_ram(11)(0)<="000";black_sprite_ram(11)(1)<="000";black_sprite_ram(11)(
2)<="000";black_sprite_ram(11)(3)<="000";black_sprite_ram(11)(4)<="000";black_sprit
e_ram(11)(5)<="000";black_sprite_ram(11)(6)<="000";black_sprite_ram(11)(7)<="000"
;black_sprite_ram(11)(8)<="000";black_sprite_ram(11)(9)<="000";black_sprite_ram(11)
(10)<="000";black_sprite_ram(11)(11)<="000";black_sprite_ram(11)(12)<="000";black
_sprite_ram(11)(13)<="000";black_sprite_ram(11)(14)<="000";black_sprite_ram(11)(15
)<="000";

black_sprite_ram(12)(0)<="000";black_sprite_ram(12)(1)<="000";black_sprite_ram(12)(
2)<="000";black_sprite_ram(12)(3)<="000";black_sprite_ram(12)(4)<="000";black_sprit
e_ram(12)(5)<="000";black_sprite_ram(12)(6)<="000";black_sprite_ram(12)(7)<="000"
;black_sprite_ram(12)(8)<="000";black_sprite_ram(12)(9)<="000";black_sprite_ram(12)
(10)<="000";black_sprite_ram(12)(11)<="000";black_sprite_ram(12)(12)<="000";black
_sprite_ram(12)(13)<="000";black_sprite_ram(12)(14)<="000";black_sprite_ram(12)(15
)<="000";

black_sprite_ram(13)(0)<="000";black_sprite_ram(13)(1)<="000";black_sprite_ram(13)(
2)<="000";black_sprite_ram(13)(3)<="000";black_sprite_ram(13)(4)<="000";black_sprit
e_ram(13)(5)<="000";black_sprite_ram(13)(6)<="000";black_sprite_ram(13)(7)<="000"
;black_sprite_ram(13)(8)<="000";black_sprite_ram(13)(9)<="000";black_sprite_ram(13)
(10)<="000";black_sprite_ram(13)(11)<="000";black_sprite_ram(13)(12)<="000";black
_sprite_ram(13)(13)<="000";black_sprite_ram(13)(14)<="000";black_sprite_ram(13)(15
)<="000";
```

```
black_sprite_ram(14)(0)<="000";black_sprite_ram(14)(1)<="000";black_sprite_ram(14)(
2)<="000";black_sprite_ram(14)(3)<="000";black_sprite_ram(14)(4)<="000";black_sprit
e_ram(14)(5)<="000";black_sprite_ram(14)(6)<="000";black_sprite_ram(14)(7)<="000"
;black_sprite_ram(14)(8)<="000";black_sprite_ram(14)(9)<="000";black_sprite_ram(14)
(10)<="000";black_sprite_ram(14)(11)<="000";black_sprite_ram(14)(12)<="000";black
_sprite_ram(14)(13)<="000";black_sprite_ram(14)(14)<="000";black_sprite_ram(14)(15
)<="000";

black_sprite_ram(15)(0)<="000";black_sprite_ram(15)(1)<="000";black_sprite_ram(15)(
2)<="000";black_sprite_ram(15)(3)<="000";black_sprite_ram(15)(4)<="000";black_sprit
e_ram(15)(5)<="000";black_sprite_ram(15)(6)<="000";black_sprite_ram(15)(7)<="000"
;black_sprite_ram(15)(8)<="000";black_sprite_ram(15)(9)<="000";black_sprite_ram(15)
(10)<="000";black_sprite_ram(15)(11)<="000";black_sprite_ram(15)(12)<="000";black
_sprite_ram(15)(13)<="000";black_sprite_ram(15)(14)<="000";black_sprite_ram(15)(15
)<="000";


    end if;
 end process;

------------------------------------------------------------------------------------------
-------------------------------------------TRY-------------------------------------------
------------------------------------------------------------------------------------------
peripheral: process(clk)
  begin
     if rising_edge(clk) then
                if avs_s1_reset_n = '0' then
                        avs_s1_readdata <= (others => '0');
                        display_address <= (others => '0');
                else
          if avs_s1_chipselect = '1' then
             if avs_s1_write = '1' then
                 if avs_s1_address = "0000000" then
                                        rhs <= avs_s1_writedata(9 downto 0);
--                                        rvs <= avs_s1_writedata(25 downto 16);
                                 elsif avs_s1_address = "0000001" then
                                        rvs <= avs_s1_writedata(9 downto 0);

                 elsif avs_s1_address = "0000010" then
```

```
                                        RAM_temp(6)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0000011" then
                                        RAM_temp(7)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0000100" then
                                        RAM_temp(8)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0000101" then
                                        RAM_temp(9)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0000110" then
                                        RAM_temp(10)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0000111" then
                                        RAM_temp(11)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001000" then
                                        RAM_temp(12)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001001" then
                                        RAM_temp(13)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001010" then
                                        RAM_temp(14)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001011" then
                                        RAM_temp(15)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001100" then
                                        RAM_temp(16)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001101" then
                                        RAM_temp(17)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001110" then
                                        RAM_temp(18)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                                elsif avs_s1_address = "0001111" then
                                        RAM_temp(19)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010000" then
                              RAM_temp(20)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010001" then
                              RAM_temp(21)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010010" then
                              RAM_temp(22)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010011" then
                              RAM_temp(23)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010100" then
                              RAM_temp(24)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010101" then
                              RAM_temp(25)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010110" then
                              RAM_temp(26)(35 downto 4) <= avs_s1_writedata(31 downto
0);
                        elsif avs_s1_address = "0010111" then
                              RAM_temp(27)(35 downto 4) <= avs_s1_writedata(31 downto
0);

                        elsif avs_s1_address = "0011001" then
                              ghost1_rhs <= avs_s1_writedata(9 downto 0);
                        elsif avs_s1_address = "0011010" then
                              ghost1_rvs <= avs_s1_writedata(9 downto 0);
--                              ghost1_rvs <= avs_s1_writedata(25 downto 16);
                        elsif avs_s1_address = "0011011" then
                              ghost2_rhs <= avs_s1_writedata(9 downto 0);
                        elsif avs_s1_address = "0011100" then
                              ghost2_rvs <= avs_s1_writedata(9 downto 0);
--                              ghost2_rvs <= avs_s1_writedata(25 downto 16);
                 elsif avs_s1_address = "0011101" then
                   score_ram_temp_2 <= avs_s1_writedata(3 downto 0);
                 elsif avs_s1_address = "0011110" then
                   score_ram_temp_1 <= avs_s1_writedata(3 downto 0);
```

```
                      elsif avs_s1_address = "0011111" then
              pacman_direction_temp <= avs_s1_writedata(2 downto 0);
                      elsif avs_s1_address = "0100000" then
              record_ram_temp_2 <= avs_s1_writedata(3 downto 0);
            elsif avs_s1_address = "0100001" then
              record_ram_temp_1 <= avs_s1_writedata(3 downto 0);
            end if;
          end if;
        end if;
        if EndOfLine='1' and EndOfField='1'then
                  pacman_x<=rhs;
          pacman_y<=rvs;
                  ghost_1_x<=ghost1_rhs;
                  ghost_1_y<=ghost1_rvs;
                  ghost_2_x<=ghost2_rhs;
                  ghost_2_y<=ghost2_rvs;
                  RAM<=RAM_temp;
                  score_ram_2<=score_ram_temp_2;
                  score_ram_1<=score_ram_temp_1;
                  record_ram_2<=record_ram_temp_2;
                  record_ram_1<=record_ram_temp_1;
                  pacman_direction <= pacman_direction_temp;
        end if;
      end if;
     end if;
end process peripheral;
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------

       process (clk)
       begin
             if clk'event and clk = '1' then
 wall_square_state_from_RAM <=
RAM(conv_integer(square_y))(conv_integer(square_x));

             end if;
       end process;

  HCounter : process (clk_25, reset_n)
```

```
begin
  if reset_n = '1' then
    Hcount <= (others => '0');
  elsif clk_25'event and clk_25 = '1' then
    if 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 (clk_25, reset_n)
begin
  if reset_n = '1' then
    Vcount <= (others => '0');
  elsif clk_25'event and clk_25 = '1' then
    if EndOfLine = '1' then
      if EndOfField = '1' then
        Vcount <= (others => '0');
      else
        Vcount <= Vcount + 1;
            visible_ypos <= Vcount - VSYNC - VBACK_PORCH;
                square_sprite_y <= visible_ypos(3 downto 0);
                if square_y /= visible_ypos(9 downto 4) then
                          square_y <= visible_ypos(9 downto 4);

                end if;
      end if;
    end if;
  end if;
end process VCounter;

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

HSyncGen : process (clk_25, reset_n)
begin
  if reset_n = '1' then
```

```
    vga_hsync <= '1';
  elsif clk_25'event and clk_25 = '1' then
    if EndOfLine = '1' then
      vga_hsync <= '1';
    elsif Hcount = HSYNC - 1 then
      vga_hsync <= '0';
    end if;
  end if;
end process HSyncGen;

HBlankGen : process (clk_25, reset_n)
begin
  if reset_n = '1' then
    vga_hblank <= '1';
  elsif clk_25'event and clk_25 = '1' then
    if 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 (clk_25, reset_n)
begin
  if reset_n = '1' then
    vga_vsync <= '1';
  elsif clk_25'event and clk_25 = '1' then
    if 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 (clk_25, reset_n)
begin
```

```
    if reset_n = '1' then
      vga_vblank <= '1';
    elsif clk_25'event and clk_25 = '1' then
      if 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;


  VideoOut: process (clk_25, reset_n)
  begin
    if reset_n = '1' then
      VGA_R <= "0000000000";
      VGA_G <= "0000000000";
      VGA_B <= "0000000000";
    elsif clk_25'event and clk_25 = '1' then         --it is here that we determine which
sprites take precedence over each other
                                                               --this is really important
stuff!
          if draw_pacman_0 = '1' then

        if pacman_direction = "000" then---up
        VGA_R <=
color_ram(conv_integer(pacman_sprite_up_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(29
downto
20);
                VGA_G <=
color_ram(conv_integer(pacman_sprite_up_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(19
downto
10);
                VGA_B <=
color_ram(conv_integer(pacman_sprite_up_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(9
```

```
downto
0);
    -----------------------------------------------------------
        elsif pacman_direction = "010" then---down
        VGA_R <=
color_ram(conv_integer(pacman_sprite_down_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(29
downto
20);
                VGA_G <=
color_ram(conv_integer(pacman_sprite_down_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(19
downto
10);
                VGA_B <=
color_ram(conv_integer(pacman_sprite_down_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(9
downto
0);
        elsif pacman_direction = "011" then ---left
                  VGA_R <=
color_ram(conv_integer(pacman_sprite_left_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(29
downto
20);
                VGA_G <=
color_ram(conv_integer(pacman_sprite_left_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(19
downto
10);
                VGA_B <=
color_ram(conv_integer(pacman_sprite_left_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(9
downto
0);
--    -----------------------------------------------------------
-----------------------------------------------------------
        elsif pacman_direction = "001" then ---right
        VGA_R <=
color_ram(conv_integer(pacman_sprite_right_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(29
downto
20);
                VGA_G <=
color_ram(conv_integer(pacman_sprite_right_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(19
downto
10);
                VGA_B <=
color_ram(conv_integer(pacman_sprite_right_ram(conv_integer(visible_ypos)-
conv_integer(pacman_y))(conv_integer(visible_xpos)-conv_integer(pacman_x))))(9
downto
0);
     ----------------------------------------------------------
                end if;

      elsif draw_ghost_1 = '1' then
        VGA_R <=
color_ram(conv_integer(ghost_sprite_1_ram(conv_integer(visible_ypos)-
conv_integer(ghost_1_y))(conv_integer(visible_xpos)-conv_integer(ghost_1_x))))(29
downto
20);
                VGA_G <=
color_ram(conv_integer(ghost_sprite_1_ram(conv_integer(visible_ypos)-
conv_integer(ghost_1_y))(conv_integer(visible_xpos)-conv_integer(ghost_1_x))))(19
downto
10);
                VGA_B <=
color_ram(conv_integer(ghost_sprite_1_ram(conv_integer(visible_ypos)-
conv_integer(ghost_1_y))(conv_integer(visible_xpos)-conv_integer(ghost_1_x))))(9
downto
0);
      elsif draw_ghost_2 = '1' then
        VGA_R <=
color_ram(conv_integer(ghost_sprite_1_ram(conv_integer(visible_ypos)-
conv_integer(ghost_2_y))(conv_integer(visible_xpos)-conv_integer(ghost_2_x))))(29
downto
20);
                VGA_G <=
color_ram(conv_integer(ghost_sprite_1_ram(conv_integer(visible_ypos)-
conv_integer(ghost_2_y))(conv_integer(visible_xpos)-conv_integer(ghost_2_x))))(19
downto
10);
                VGA_B <=
color_ram(conv_integer(ghost_sprite_1_ram(conv_integer(visible_ypos)-
conv_integer(ghost_2_y))(conv_integer(visible_xpos)-conv_integer(ghost_2_x))))(9
downto
0);



      elsif vga_hblank = '0' and vga_vblank ='0' then
                        if print_char = '1' then --if we should be printing some number or text
instead of the
maze:

                            VGA_R <= (others =>
current_char_sprite_from_RAM(conv_integer(square_sprite_y))(conv_integer(square_sp
rite_x)));
                          VGA_G <= (others =>
current_char_sprite_from_RAM(conv_integer(square_sprite_y))(conv_integer(square_sp
rite_x)));
                          VGA_B <= (others =>
current_char_sprite_from_RAM(conv_integer(square_sprite_y))(conv_integer(square_sp
rite_x)));

                        elsif show_wall_square = '1' then
                           VGA_R <=
color_ram(conv_integer(square_sprite_ram(conv_integer(square_sprite_y))(conv_integer
(square_sprite_x))))(29 downto
20);
                           VGA_G <=
color_ram(conv_integer(square_sprite_ram(conv_integer(square_sprite_y))(conv_integer
(square_sprite_x))))(19 downto
10);
                           VGA_B <=
color_ram(conv_integer(square_sprite_ram(conv_integer(square_sprite_y))(conv_integer
(square_sprite_x))))(9 downto
0);
            else
                           VGA_R <= "0000000000";
```

```
              VGA_G <= "0000000000";
                  VGA_B <= "0000000000";
           end if;
     else
               VGA_R <= "0000000000";
       VGA_G <= "0000000000";
           VGA_B <= "0000000000";
     end if;


   end if;
  end process VideoOut;

  VGA_CLK <= clk_25;
  VGA_HS <= not vga_hsync;
  VGA_VS <= not vga_vsync;
  VGA_SYNC <= '0';
  VGA_BLANK <= not (vga_hsync or vga_vsync);

end rtl;
```

4.1.4 tone_generator// The Audio Controller

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

entity tone_generator is

  port (
    avs_s1_clk        : in  std_logic;
    avs_s1_reset_n    : in  std_logic;
    avs_s1_read       : in  std_logic;
    avs_s1_write      : in  std_logic;
    avs_s1_chipselect : in  std_logic;
    avs_s1_address    : in  std_logic;
    avs_s1_readdata   : out unsigned(15 downto 0);
    avs_s1_writedata  : in  unsigned(15 downto 0);
        frq                  : out unsigned(15 downto 0);
        mod_depth  : out unsigned(3 downto 0)
    );
```

```
end tone_generator;

architecture rtl of tone_generator is
        signal frq_reg: unsigned(15 downto 0);
        signal mod_depth_reg:unsigned(3 downto 0);
        signal cnt:unsigned(27 downto 0):=x"0000000";
    signal x: std_logic:='0';
        signal y: std_logic:='0';
        signal flag: unsigned(2 downto 0):="011";
        signal clk : std_logic:='0';

begin

clk <= avs_s1_clk;

peripheral: process(clk)
  begin
     if rising_edge(clk) then
                if avs_s1_reset_n = '0' then
                        avs_s1_readdata <= (others => '0');
                else
          if avs_s1_chipselect = '1' then
             if avs_s1_write = '1' then
                 if avs_s1_address = '0' then
                                        flag <= avs_s1_writedata(2 downto 0);
                 end if;
             end if;
          end if;
       end if;
     end if;
end process peripheral;

  process (clk)
  begin
    if rising_edge(clk) then
        --if reset_n = '0' then
        if flag="000" then
      if (x='0' and y='0') then
        frq_reg <= x"0951";
                mod_depth_reg <=x"7";
                cnt<=x"0000000";
                frq<=frq_reg;
            mod_depth<=mod_depth_reg;
            y <= '1';
                end if;

         if(x='0' and y='1') then
            cnt <= cnt +1;
                        if(cnt = x"2FFFFFF") then
                        frq_reg <= x"0597";
                   mod_depth_reg <=x"A";
                    cnt<=x"0000000";
                frq<=frq_reg;
            mod_depth<=mod_depth_reg;
                x <= '1';
                        end if;
        end if;

        if(x='1' and y='1') then
            cnt <= cnt +1;
                        if(cnt = x"1FFFFFF") then
                frq_reg <= x"0345";
                   mod_depth_reg <=x"A";
                    cnt<=x"0000000";
                frq<=frq_reg;
            mod_depth<=mod_depth_reg;
             y<='0';
            end if;
        end if;

        if(x='1' and y='0') then
                cnt <= cnt +1;
                        if(cnt = x"2FFFFFF") then

                x<='0';
                y<='0';
                cnt<=x"0000000";
                end if;
        end if;
```

```
  elsif flag="001" then
 if (x='0' and y='0') then
  frq_reg <= x"0951";
        mod_depth_reg <=x"7";
        cnt<=x"0000000";
        frq<=frq_reg;
    mod_depth<=mod_depth_reg;
    y <= '1';
        end if;

   if(x='0' and y='1') then
    cnt <= cnt +1;
            if(cnt = x"0EFFFFF") then
            frq_reg <= x"0871";
          mod_depth_reg <=x"A";
           cnt<=x"0000000";
        frq<=frq_reg;
    mod_depth<=mod_depth_reg;
        x <= '1';
            end if;
  end if;

  if(x='1' and y='1') then
    cnt <= cnt +1;
            if(cnt = x"0EFFFFF") then
        frq_reg <= x"0751";
          mod_depth_reg <=x"A";
           cnt<=x"0000000";
        frq<=frq_reg;
    mod_depth<=mod_depth_reg;
     y<='0';
    end if;
  end if;

  if(x='1' and y='0') then
        cnt <= cnt +1;
            if(cnt = x"0000FFF") then

        x<='0';
        y<='0';
```

```
                cnt<=x"0000000";
                end if;
        end if;

elsif flag="010" then
 if (x='0' and y='0') then
        frq_reg <= x"0971";
                mod_depth_reg <=x"A";
                cnt<=x"0000000";
                frq<=frq_reg;
            mod_depth<=mod_depth_reg;
            y <= '1';
                end if;

         if(x='0' and y='1') then
            cnt <= cnt +1;
                        if(cnt = x"08FFFFF") then
                        frq_reg <= x"0000";
                   mod_depth_reg <=x"0";
                    cnt<=x"0000000";
                frq<=frq_reg;
            mod_depth<=mod_depth_reg;
            x<='0';
                y<='0';
                        end if;
        end if;

 elsif flag="011" then

     if (x='0' and y='0') then
        frq_reg <= x"0751";
                mod_depth_reg <=x"7";
                cnt<=x"0000000";
                frq<=frq_reg;
            mod_depth<=mod_depth_reg;
            y <= '1';
                end if;

         if(x='0' and y='1') then
            cnt <= cnt +1;
                        if(cnt = x"0FFFFFF") then
                frq_reg <= x"0697";
             mod_depth_reg <=x"A";
              cnt<=x"0000000";
           frq<=frq_reg;
        mod_depth<=mod_depth_reg;
           x <= '1';
                end if;
      end if;

      if(x='1' and y='1') then
        cnt <= cnt +1;
                if(cnt = x"0FFFFFF") then
           frq_reg <= x"0545";
             mod_depth_reg <=x"A";
              cnt<=x"0000000";
           frq<=frq_reg;
        mod_depth<=mod_depth_reg;
         y<='0';
        end if;
      end if;

      if(x='1' and y='0') then
           cnt <= cnt +1;
                if(cnt = x"0FFFFFF") then
           x<='0';
           y<='0';
           cnt<=x"0000000";
           end if;
      end if;
    else
       x<='0';
            y<='0';
       frq_reg <= x"0000";
             mod_depth_reg <=x"0";
           frq<=frq_reg;
        mod_depth<=mod_depth_reg;
    end if;
      end if;
  end process;
end rtl;
```

4.1.5 lab3_audio.vhd // The Top Level Entity

```
--
-- DE2 top-level module that includes the simple audio component
--
-- 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;

use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity lab3_audio 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):=b"1111111";
    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_audio;

architecture datapath of lab3_audio is

  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;
----
component nios_system

        port (
               clk : IN STD_LOGIC;
               reset_n : IN STD_LOGIC;

             HEX0_from_the_LED7_inst : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);
                HEX1_from_the_LED7_inst : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);
                  HEX2_from_the_LED7_inst : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);
              HEX3_from_the_LED7_inst : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);
          HEX4_from_the_LED7_inst : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);

              PS2_Clk_to_the_de2_ps2_inst : IN STD_LOGIC;
              PS2_Data_to_the_de2_ps2_inst : IN STD_LOGIC;
```

```
                 SRAM_ADDR_from_the_sram : OUT STD_LOGIC_VECTOR (17
DOWNTO 0);
                 SRAM_CE_N_from_the_sram : OUT STD_LOGIC;
                 SRAM_DQ_to_and_from_the_sram : INOUT STD_LOGIC_VECTOR (15
DOWNTO 0);
                 SRAM_LB_N_from_the_sram : OUT STD_LOGIC;
                 SRAM_OE_N_from_the_sram : OUT STD_LOGIC;
                 SRAM_UB_N_from_the_sram : OUT STD_LOGIC;
                 SRAM_WE_N_from_the_sram : OUT STD_LOGIC;
     key_status_to_the_switch_reader_inst : IN STD_LOGIC_VECTOR (15 DOWNTO
0);


            -- the_tone_generator_inst
              frq_from_the_tone_generator_inst : OUT STD_LOGIC_VECTOR (15
DOWNTO 0);
              mod_depth_from_the_tone_generator_inst : OUT STD_LOGIC_VECTOR (3
DOWNTO
0)
            );
end component;

component sdram_pll
port(inclk0:in std_logic;
c0:out std_logic;
c1:out std_logic);
end component;

signal BA:std_logic_vector(1 downto 0);
signal DQM:std_logic_vector(1 downto 0);
signal pll_c1:std_logic;


  signal audio_clock : std_logic_vector(1 downto 0) := "00";
  signal audio_request : std_logic;
  signal frq:std_logic_vector (15 downto 0);
  signal mod_depth:std_logic_vector (3 downto 0);

  signal counter : std_logic_vector(15 downto 0) := "0000000000000000";
  signal reset_n : std_logic :='0';
  signal clk25 : std_logic := '0';
```

```
begin

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

  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
  );

DRAM_BA_1<=BA(1);
DRAM_BA_0<=BA(0);
DRAM_UDQM<=DQM(1);
DRAM_LDQM<=DQM(0);

  process (CLOCK_50)
  begin
    if CLOCK_50'event and CLOCK_50 = '1' then
      clk25 <= not clk25;
    end if;
  end process;

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



V5:entity work.de2_wm8731_audio port map(
    clk         => audio_clock(1),--  Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
    reset_n => '1',
    test_mode => '1',--    Audio CODEC controller test mode
    audio_request => audio_request,--    Audio controller request new data
    data => "0000000000000000",

    -- Audio interface signals
    AUD_ADCLRCK => AUD_ADCLRCK,  --    Audio CODEC ADC LR Clock
    AUD_ADCDAT  => AUD_ADCDAT,  --    Audio CODEC ADC Data
    AUD_DACLRCK => AUD_DACLRCK  , --    Audio CODEC DAC LR Clock
    AUD_DACDAT  => AUD_DACDAT, --    Audio CODEC DAC Data
    AUD_BCLK    => AUD_BCLK  , --    Audio CODEC Bit-Stream Clock
        frq_in                    => frq,
        mod_depth_in        => mod_depth
  );

nios: entity work.nios_system
    port map(

      frq_from_the_tone_generator_inst => frq,
      mod_depth_from_the_tone_generator_inst => mod_depth,
      clk => pll_c1,
      reset_n => '1',
      PS2_Clk_to_the_de2_ps2_inst =>PS2_CLK,
      PS2_Data_to_the_de2_ps2_inst =>PS2_DAT,
      key_status_to_the_switch_reader_inst => SW(15 downto 0),
      HEX0_from_the_LED7_inst => HEX0,--: OUT STD_LOGIC_VECTOR (6
DOWNTO 0);
      HEX1_from_the_LED7_inst => HEX1,--: OUT STD_LOGIC_VECTOR (6
DOWNTO 0);
      HEX2_from_the_LED7_inst => HEX2,--: OUT STD_LOGIC_VECTOR (6
DOWNTO 0);
      HEX3_from_the_LED7_inst => HEX3,--: OUT STD_LOGIC_VECTOR (6
DOWNTO 0);
      HEX4_from_the_LED7_inst => HEX4,--: OUT STD_LOGIC_VECTOR (6
DOWNTO 0);
```

```
    reset_n_to_the_de2_vga_raster_inst                 => '0',
    CLK_25_to_the_de2_vga_raster_inst         => clk25,
  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_de2_vga_raster_inst                 => VGA_CLK,
    VGA_HS_from_the_de2_vga_raster_inst                 => VGA_HS,
    VGA_VS_from_the_de2_vga_raster_inst         => VGA_VS,
    VGA_BLANK_from_the_de2_vga_raster_inst         => VGA_BLANK,
   VGA_SYNC_from_the_de2_vga_raster_inst         => VGA_SYNC,
   VGA_R_from_the_de2_vga_raster_inst         => VGA_R,
    VGA_G_from_the_de2_vga_raster_inst          => VGA_G,
    VGA_B_from_the_de2_vga_raster_inst         => VGA_B

  );


neg_3ns:sdram_pll PORT MAP(CLOCK_50,DRAM_CLK,pll_c1);


 LEDG     <= (others => '0');
 LEDR     <= (others => '0');
 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';
```

```
  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';

  -- Set all bidirectional ports to tri-state
  DRAM_DQ     <= (others => 'Z');
  FL_DQ       <= (others => 'Z');
  OTG_DATA    <= (others => 'Z');
  LCD_DATA    <= (others => 'Z');
  SD_DAT      <= 'Z';
  ENET_DATA   <= (others => 'Z');
  GPIO_0      <= (others => 'Z');
  GPIO_1      <= (others => 'Z');

end datapath;
```

4.2 Software

4.2.1 hello_world.c // The Game Logic

```
/*
 * "Hello World" example.
 *
 * This example prints 'Hello from Nios II' to the STDOUT stream. It runs on
 * the Nios II 'standard', 'full_featured', 'fast', and 'low_cost' example
 * designs. It runs with or without the MicroC/OS-II RTOS and requires a STDOUT
 * device in your system's hardware.
 * The memory footprint of this hosted application is ~69 kbytes by default
 * using the standard reference design.
 *
 * For a reduced footprint version of this template, and an explanation of how
 * to reduce the memory footprint for a given application, see the
 * "small_hello_world" template.
 *
 */

#include <io.h>
#include <system.h>
#include <stdio.h>
#include <alt_types.h>
#include <sys/time.h>
#include <sys/alt_timestamp.h>
#include <stdlib.h>
#include <time.h>

#define NONE 0
#define UP 1
#define RIGHT 2
#define DOWN 3
#define LEFT 4

#define SPACE 0
#define WALL 1
#define GHOST 2
#define TAIL 3

#define x_offset 48
#define y_offset 81
```

```
int maze_temp[24][34];
int maze_to_ram[22][32];
int pac_x;
int pac_y;
int ghost1_x;
int ghost1_y;
int ghost2_x;
int ghost2_y;
int score;
int record;
int counter;
int step;
int direction;
int ghost1_x_flag=1;
int ghost1_y_flag=-1;
int ghost2_x_flag=-1;
int ghost2_y_flag=1;
int audio_counter=0;
int freezing=0;

#define GHOST1_SPEED 1<<8
#define GHOST2_SPEED 1<<7
#define PAC_SPEED 1<<12
#define AUDIO_SPEED 1<<11
#define INIT_POWERUP_COUNTER 350

#define PAC_OFFSET 0 //adress=0,1
#define MAZE_OFFSET 8 //adress=2-23
#define GHOST_OFFSET 100 //adress=25-28
#define SCORE_OFFSET 116 //adress=29,30
#define DIRECTION_OFFSET 124 //adress=31
#define RECORD_OFFSET 128 //adress=32,33

#define IOWR_PAC(num, position) \
  IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+PAC_OFFSET, (num)*4,
position)
#define IOWR_MAZE(line, data) \
  IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+MAZE_OFFSET, (line)*4,
data)
#define IOWR_GHOST(num, position) \
 IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+GHOST_OFFSET, (num)*4,
position)

void initialize()
{
   //initialize the maze
   int i,j;
   for(i=0;i<24;i++){
      for(j=0;j<34;j++){
         if(i==0||j==0||i==23||j==33)
            maze_temp[i][j]=1;
         else
            maze_temp[i][j]=0;
         //maze_to_ram is the output maze with only 0 and 1, maze_temp is the maze
only in C code
      }
   }

   //initialize the position of the pacman
   pac_x=0;
   pac_y=0;

   //initialize the position of ghosts, randomly
   srand((unsigned)time(NULL));
   ghost1_x=rand()%32+1;
   ghost1_y=rand()%22+1;
   ghost2_x=rand()%32+1;
   ghost2_y=rand()%22+1;

   //initialize the score
   counter=0;
   score=0;
   IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+SCORE_OFFSET,0,score/1
0);
   IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+SCORE_OFFSET,4,score%
10);

   direction=NONE;

   audio_counter=0;
   IOWR_16DIRECT(TONE_GENERATOR_INST_BASE, 0, 0); //starting audio
```

```
    step=0;
    freezing=0;
}

void output(int a[24][34])
{
    //change the maze_temp into the maze which is made of '0' or '1'
    //IOWR_MAZE
    int i,j,k,r;
    int temp=0;
    for(i=1;i<23;i++){
        for(j=1;j<33;j++){
            if(maze_temp[i][j]%2==0)
                maze_to_ram[i-1][j-1]=0;
            else
                maze_to_ram[i-1][j-1]=1;
            //printf("%d",maze_to_ram[i-1][j-1]);
            //printf("%d",maze_temp[i][j]);
        }
        //printf("\n");
    }
    //printf("\n");

    for(k=0;k<22;k++){
        temp=0;
        for(r=31;r>-1;r--)
            temp=temp*2+maze_to_ram[k][r]; //convert to int
        //printf("%d\n",temp);
        IOWR_MAZE(k,temp);
    }
//    IOWR_MAZE(2,1024*128*128);
}

void calmaze()
{
    int counter_TAIL=0;
    int cal_direction[24][34];
    int i,j;

    for(i=0;i<24;i++){
        for(j=0;j<34;j++){
            if(maze_temp[i][j]==TAIL){ //TAIL to WALL
                maze_temp[i][j]=WALL;
                counter_TAIL++;
            }
//            else if(counter_TAIL!=0){
//                if(maze_temp[i][j]==SPACE||maze_temp[i][j]==GHOST) //mark else
//                    maze_temp[i][j]=5;
//            }
        }
    }

    if(counter_TAIL!=0){ //need to calculate
        for(i=0;i<24;i++){
            for(j=0;j<34;j++){
                if(maze_temp[i][j]==SPACE||maze_temp[i][j]==GHOST) //mark else
                    maze_temp[i][j]=5;
            }
        }
        if(maze_temp[ghost1_y][ghost1_x]%2!=0){ //it hasn't been visited
            maze_temp[ghost1_y][ghost1_x]=2; //visit it and start from it
            int i=ghost1_y;
            int j=ghost1_x;
            while(cal_direction[i][j]!=5){
                if(maze_temp[i-1][j]==5){
                    i=i-1;
                    cal_direction[i][j]=UP;
                    maze_temp[i][j]=SPACE;
                    //counter++;
                }
                else if(maze_temp[i][j+1]==5){
                    j=j+1;
                    cal_direction[i][j]=RIGHT;
                    maze_temp[i][j]=SPACE;
                    //counter++;
                }
                else if(maze_temp[i+1][j]==5){
                    i=i+1;
                    cal_direction[i][j]=DOWN;
                    maze_temp[i][j]=SPACE;
                    //counter++;
                }
                else if(maze_temp[i][j-1]==5){
                    j=j-1;
                    cal_direction[i][j]=LEFT;
                    maze_temp[i][j]=SPACE;
                    //counter++;
                }
                else{
                    if(i==ghost1_y&&j==ghost1_x) //calculation done
                        cal_direction[i][j]=5;
                    else if(cal_direction[i][j]==UP)
                        i=i+1;
                    else if(cal_direction[i][j]==RIGHT)
                        j=j-1;
                    else if(cal_direction[i][j]==DOWN)
                        i=i-1;
                    else if(cal_direction[i][j]==LEFT)
                        j=j+1;
                }
            }
        }

        if(maze_temp[ghost2_y][ghost2_x]%2!=0){ //it hasn't been visited
            maze_temp[ghost2_y][ghost2_x]=2; //visit it and start from it
            int i=ghost2_y;
            int j=ghost2_x;
            while(cal_direction[i][j]!=5){
                if(maze_temp[i-1][j]==5){
                    i=i-1;
                    cal_direction[i][j]=UP;
                    maze_temp[i][j]=SPACE;
                    //counter++;
                }
                else if(maze_temp[i][j+1]==5){
                    j=j+1;
                    cal_direction[i][j]=RIGHT;
                    maze_temp[i][j]=SPACE;
                    //counter++;
                }
                else if(maze_temp[i+1][j]==5){
                    i=i+1;
```

```
                cal_direction[i][j]=DOWN;
                maze_temp[i][j]=SPACE;
                //counter++;
            }
            else if(maze_temp[i][j-1]==5){
                j=j-1;
                cal_direction[i][j]=LEFT;
                maze_temp[i][j]=SPACE;
                //counter++;
            }
            else{
                if(i==ghost2_y&&j==ghost2_x) //calculation done
                    cal_direction[i][j]=5;
                else if(cal_direction[i][j]==UP)
                    i=i+1;
                else if(cal_direction[i][j]==RIGHT)
                    j=j-1;
                else if(cal_direction[i][j]==DOWN)
                    i=i-1;
                else if(cal_direction[i][j]==LEFT)
                    j=j+1;
            }
        }
    }
    IOWR_16DIRECT(TONE_GENERATOR_INST_BASE, 0, 2);
    audio_counter=0;
}

for(i=0;i<24;i++){
    for(j=0;j<34;j++){
        if(maze_temp[i][j]==5){ //5 to WALL
            maze_temp[i][j]=WALL;
        }
    }
}
//calculate and output score
for(i=1;i<23;i++){
    for(j=1;j<33;j++){
        if(maze_temp[i][j]==WALL){ //5 to WALL
            counter++;
        }
        }
    }
    score=counter*100/704;
    printf("score is %d\n",score);
    printf("record is %d\n",record);
    if(record<score){
        record=score;
        IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+RECORD_OFFSET,0,sco
re/10);
        IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+RECORD_OFFSET,4,sco
re%10);
    }
    counter=0;
    IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+SCORE_OFFSET,0,score/1
0);
    IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+SCORE_OFFSET,4,score%
10);
}

int main()
{
    int pac_counter=0;
    int ghost1_counter=0;
    int ghost2_counter=0;
    int sub1_counter=0;
    int sub2_counter=0;

    int pac_speed=PAC_SPEED;
    int ghost1_speed=GHOST1_SPEED;
    int ghost2_speed=GHOST2_SPEED;

    unsigned char code;
    unsigned char precode;

    initialize(); //start

    for(;;){

        if(freezing==0){

        pac_counter++;
        ghost1_counter++;
        ghost2_counter++;
        audio_counter++;

        pac_counter%=pac_speed;
        ghost1_counter%=ghost1_speed;
        ghost2_counter%=ghost2_speed;
        audio_counter%=AUDIO_SPEED;

        if(audio_counter==0){
          IOWR_16DIRECT(TONE_GENERATOR_INST_BASE, 0, 5);
        }

        if(ghost1_counter==0){ //ghosts move
          sub1_counter++;
          sub1_counter%=16;

if(maze_temp[ghost1_y][ghost1_x+1]==TAIL||maze_temp[ghost1_y][ghost1_x-
1]==TAIL||maze_temp[ghost1_y+1][ghost1_x]==TAIL||maze_temp[ghost1_y-
1][ghost1_x]==TAIL||maze_temp[ghost1_y-
1][ghost1_x+1]==TAIL||maze_temp[ghost1_y+1][ghost1_x-
1]==TAIL||maze_temp[ghost1_y+1][ghost1_x+1]==TAIL||maze_temp[ghost1_y-
1][ghost1_x-1]==TAIL)
          //initialize(); //game over
          freezing=1;
          else{


if(maze_temp[ghost1_y][ghost1_x+1]==WALL||maze_temp[ghost1_y][ghost1_x+1]==T
AIL||maze_temp[ghost1_y][ghost1_x+1]==5)
            ghost1_x_flag=-1;

if(maze_temp[ghost1_y][ghost1_x-1]==WALL||maze_temp[ghost1_y][ghost1_x-
1]==TAIL||maze_temp[ghost1_y][ghost1_x-1]==5)
            ghost1_x_flag=1;

if(maze_temp[ghost1_y-1][ghost1_x]==WALL||maze_temp[ghost1_y-
1][ghost1_x]==TAIL||maze_temp[ghost1_y-1][ghost1_x]==5)
            ghost1_y_flag=1;

if(maze_temp[ghost1_y+1][ghost1_x]==WALL||maze_temp[ghost1_y+1][ghost1_x]==T
AIL||maze_temp[ghost1_y+1][ghost1_x]==5)
                ghost1_y_flag=-1;

if(maze_temp[ghost1_y][ghost1_x+1]!=WALL&&maze_temp[ghost1_y][ghost1_x-
1]!=WALL&&maze_temp[ghost1_y-
1][ghost1_x]!=WALL&&maze_temp[ghost1_y+1][ghost1_x]!=WALL){

if((maze_temp[ghost1_y+1][ghost1_x+1]==WALL&&ghost1_x_flag==1&&ghost1_y_fl
ag==1)||(maze_temp[ghost1_y+1][ghost1_x+1]==GHOST&&ghost1_x_flag==1&&ghos
t1_y_flag==1)){
                   ghost1_x_flag=-1;
                   ghost1_y_flag=-1;
                }

if((maze_temp[ghost1_y-1][ghost1_x-1]==WALL&&ghost1_x_flag==-
1&&ghost1_y_flag==-1)||(maze_temp[ghost1_y-1][ghost1_x-
1]==GHOST&&ghost1_x_flag==-1&&ghost1_y_flag==-1)){
                   ghost1_x_flag=1;
                   ghost1_y_flag=1;
                }

if((maze_temp[ghost1_y-
1][ghost1_x+1]==WALL&&ghost1_x_flag==1&&ghost1_y_flag==-
1)||(maze_temp[ghost1_y-
1][ghost1_x+1]==GHOST&&ghost1_x_flag==1&&ghost1_y_flag==-1)){
                   ghost1_x_flag=-1;
                   ghost1_y_flag=1;
                }

if((maze_temp[ghost1_y+1][ghost1_x-1]==WALL&&ghost1_x_flag==-
1&&ghost1_y_flag==1)||(maze_temp[ghost1_y+1][ghost1_x-
1]==GHOST&&ghost1_x_flag==-1&&ghost1_y_flag==1)){
                   ghost1_x_flag=1;
                   ghost1_y_flag=-1;
                }
             }

             if(sub1_counter==0){
                ghost1_x=ghost1_x+ghost1_x_flag;
                ghost1_y=ghost1_y+ghost1_y_flag;
                maze_temp[ghost1_y-ghost1_y_flag][ghost1_x-ghost1_x_flag]=SPACE;
                maze_temp[ghost1_y][ghost1_x]=GHOST;
            }
                //IOWR_GHOST(1,ghost1_x*16+x_offset+(ghost1_y*16+y_offset)<<16);
                IOWR_GHOST(0,ghost1_x*16+ghost1_x_flag*sub1_counter+x_offset);
                IOWR_GHOST(1,ghost1_y*16+ghost1_y_flag*sub1_counter+y_offset);
        }
    }

    if(ghost2_counter==0){
        sub2_counter++;
        sub2_counter%=16;

if(maze_temp[ghost2_y][ghost2_x+1]==TAIL||maze_temp[ghost2_y][ghost2_x-
1]==TAIL||maze_temp[ghost2_y+1][ghost2_x]==TAIL||maze_temp[ghost2_y-
1][ghost2_x]==TAIL||maze_temp[ghost2_y-
1][ghost2_x+1]==TAIL||maze_temp[ghost2_y+1][ghost2_x-
1]==TAIL||maze_temp[ghost2_y+1][ghost2_x+1]==TAIL||maze_temp[ghost2_y-
1][ghost2_x-1]==TAIL)
            //initialize(); //game over
            freezing=1;
        else{

if(maze_temp[ghost2_y][ghost2_x+1]==WALL||maze_temp[ghost2_y][ghost2_x+1]==T
AIL||maze_temp[ghost2_y][ghost2_x+1]==5)
                ghost2_x_flag=-1;

if(maze_temp[ghost2_y][ghost2_x-1]==WALL||maze_temp[ghost2_y][ghost2_x-
1]==TAIL||maze_temp[ghost2_y][ghost2_x-1]==5)
                ghost2_x_flag=1;

if(maze_temp[ghost2_y-1][ghost2_x]==WALL||maze_temp[ghost2_y-
1][ghost2_x]==TAIL||maze_temp[ghost2_y-1][ghost2_x]==5)
                ghost2_y_flag=1;

if(maze_temp[ghost2_y+1][ghost2_x]==WALL||maze_temp[ghost2_y+1][ghost2_x]==T
AIL||maze_temp[ghost2_y+1][ghost2_x]==5)
                ghost2_y_flag=-1;

if(maze_temp[ghost2_y][ghost2_x+1]!=WALL&&maze_temp[ghost2_y][ghost2_x-
1]!=WALL&&maze_temp[ghost2_y-
1][ghost2_x]!=WALL&&maze_temp[ghost2_y+1][ghost2_x]!=WALL){
```

```
if((maze_temp[ghost2_y+1][ghost2_x+1]==WALL&&ghost2_x_flag==1&&ghost2_y_fl
ag==1)||(maze_temp[ghost2_y+1][ghost2_x+1]==GHOST&&ghost2_x_flag==1&&ghos
t2_y_flag==1)){
                    ghost2_x_flag=-1;
                    ghost2_y_flag=-1;
                }

if((maze_temp[ghost2_y-1][ghost2_x-1]==WALL&&ghost2_x_flag==-
1&&ghost2_y_flag==-1)||(maze_temp[ghost2_y-1][ghost2_x-
1]==GHOST&&ghost2_x_flag==-1&&ghost2_y_flag==-1)){
                    ghost2_x_flag=1;
                    ghost2_y_flag=1;
                }

if((maze_temp[ghost2_y-
1][ghost2_x+1]==WALL&&ghost2_x_flag==1&&ghost2_y_flag==-
1)||(maze_temp[ghost2_y-
1][ghost2_x+1]==GHOST&&ghost2_x_flag==1&&ghost2_y_flag==-1)){
                    ghost2_x_flag=-1;
                    ghost2_y_flag=1;
                }

if((maze_temp[ghost2_y+1][ghost2_x-1]==WALL&&ghost2_x_flag==-
1&&ghost2_y_flag==1)||(maze_temp[ghost2_y+1][ghost2_x-
1]==GHOST&&ghost2_x_flag==-1&&ghost2_y_flag==1)){
                    ghost2_x_flag=1;
                    ghost2_y_flag=-1;
                }
            }
            if(sub2_counter==0){
               ghost2_x=ghost2_x+ghost2_x_flag;
               ghost2_y=ghost2_y+ghost2_y_flag;
               maze_temp[ghost2_y-ghost2_y_flag][ghost2_x-ghost2_x_flag]=SPACE;
               maze_temp[ghost2_y][ghost2_x]=GHOST;
            }
            //IOWR_GHOST(2,ghost2_x*16+x_offset+(ghost2_y*16+y_offset)<<16);
            IOWR_GHOST(2,ghost2_x*16+ghost2_x_flag*sub2_counter+x_offset);
            IOWR_GHOST(3,ghost2_y*16+ghost2_y_flag*sub2_counter+y_offset);
        }
    }
    if(pac_counter==0){

        //input
        //the direction is decided by pressed botton and the last direction
        //we do not change into an opposite direction or a current direction
        //while(!IORD_8DIRECT(PS2_BASE,0));
        if(!IORD_8DIRECT(DE2_PS2_INST_BASE,0)) code=0;
        else{
            //if(precode==240||precode==224) code=0;
            //else code = IORD_8DIRECT(DE2_PS2_INST_BASE,4);
            //precode=code;
            //printf("%d\n",code);
            code = IORD_8DIRECT(DE2_PS2_INST_BASE,4);
        }

        switch(code){

        case 117: //UP BUTTON
            if(direction==UP||direction==DOWN) break;
            else{
                direction=UP;
                IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+DIRECTION_OF
FSET,0,0);
            }
            break;

        case 116: //RIGHT BUTTON
            if(direction==RIGHT||direction==LEFT) break;
            else{
                direction=RIGHT;
                IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+DIRECTION_OF
FSET,0,1);
            }
            break;

        case 114: //DOWN BUTTON
            if(direction==UP||direction==DOWN) break;
            else{
                direction=DOWN;
                IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+DIRECTION_OF
FSET,0,2);
            }
            break;

         case 107: //LEFT BUTTON
            if(direction==RIGHT||direction==LEFT) break;
            else{
               direction=LEFT;
               IOWR_32DIRECT(DE2_VGA_RASTER_INST_BASE+DIRECTION_OF
FSET,0,3);
            }
            break;

         default:
            break;
         }

         switch(direction){

         case UP:
            if(pac_y-1<0) //out of wall, do nothing
               direction=NONE;
            else
if(maze_temp[pac_y-1][pac_x]==TAIL||maze_temp[pac_y-1][pac_x]==GHOST) //eat its
own tail
or got the ghost, loose
               //initialize();
               freezing=1;
            else if(maze_temp[pac_y-1][pac_x]==WALL){ //get to the safe wall
               pac_y=pac_y-1;
               calmaze();
               direction=NONE;
            }
            else if(maze_temp[pac_y-1][pac_x]==SPACE){ //into space, keep ahead
               pac_y=pac_y-1;
               audio_counter=0;
               IOWR_16DIRECT(TONE_GENERATOR_INST_BASE, 0, 1);
               direction=UP;
               maze_temp[pac_y][pac_x]=TAIL;
            }
            break;
```

```
        case RIGHT:
            if(pac_x+1>33) //out of wall, do nothing
                direction=NONE;
            else
if(maze_temp[pac_y][pac_x+1]==TAIL||maze_temp[pac_y][pac_x+1]==GHOST) //eat
its own tail
or got the ghost, loose
                //initialize();
                freezing=1;
            else if(maze_temp[pac_y][pac_x+1]==WALL){ //get to the safe wall
                pac_x=pac_x+1;
                calmaze();
                direction=NONE;
            }
            else if(maze_temp[pac_y][pac_x+1]==SPACE){ //into space, keep ahead
                pac_x=pac_x+1;
                audio_counter=0;
                IOWR_16DIRECT(TONE_GENERATOR_INST_BASE, 0, 1);
                direction=RIGHT;
                maze_temp[pac_y][pac_x]=TAIL;
            }
            break;

        case DOWN:
            if(pac_y+1>23) //out of wall, do nothing
                direction=NONE;
            else
if(maze_temp[pac_y+1][pac_x]==TAIL||maze_temp[pac_y+1][pac_x]==GHOST) //eat
its own tail
or got the ghost, loose
                //initialize();
                freezing=1;
            else if(maze_temp[pac_y+1][pac_x]==WALL){ //get to the safe wall
                pac_y=pac_y+1;
                calmaze();
                direction=NONE;
            }
            else if(maze_temp[pac_y+1][pac_x]==SPACE){ //into space, keep ahead
                pac_y=pac_y+1;
                audio_counter=0;
                IOWR_16DIRECT(TONE_GENERATOR_INST_BASE, 0, 1);
                direction=DOWN;
                maze_temp[pac_y][pac_x]=TAIL;
            }
            break;

        case LEFT:
            if(pac_x-1<0) //out of wall, do nothing
                direction=NONE;
            else
if(maze_temp[pac_y][pac_x-1]==TAIL||maze_temp[pac_y][pac_x-1]==GHOST) //eat its
own tail
or got the ghost, loose
                //initialize();
                freezing=1;
            else if(maze_temp[pac_y][pac_x-1]==WALL){ //get to the safe wall
                pac_x=pac_x-1;
                calmaze();
                direction=NONE;
            }
            else if(maze_temp[pac_y][pac_x-1]==SPACE){ //into space, keep ahead
                pac_x=pac_x-1;
                audio_counter=0;
                IOWR_16DIRECT(TONE_GENERATOR_INST_BASE, 0, 1);
                direction=LEFT;
                maze_temp[pac_y][pac_x]=TAIL;
            }
            break;

        default:
            break;
        }
        //IOWR_PAC(0,pac_x*16+x_offset+(pac_y*16+y_offset)<<16);
        IOWR_PAC(0,pac_x*16+x_offset);
        IOWR_PAC(1,pac_y*16+y_offset);
        output(maze_temp);
    }
  }

  else{
    if(!IORD_8DIRECT(DE2_PS2_INST_BASE,0)) code=0;
```

```
        else{
            code = IORD_8DIRECT(DE2_PS2_INST_BASE,4);
            //printf("%d\n",code);
        }

        switch(code){

        case 90:
            initialize();
        }
    }

    }
    return 0;
}
```