CSEE 4840
EMBEDDED SYSTEMS

"Whac-A-Mole" Project Report

Spring 2016

Group Members

- *Georgios Charitos (gc2662)*
- *Aditya Bagri (aab2234)*
- *Astha Agrawal (aa3755)*
- *Jai Sharma (js4773)*

Due date: 5-12-16

Table of Contents:

1. Overview:

As a part of the course CSEE 4840, we implemented our final project, the game "Whac – A – Mole". It is a common game for children and the working of the game is straightforward: Hit as many moles as you can with the hammer. To design this game, we used the Cyclone V board with the Altera FPGA embedded in it. The other components include the VGA Screen display, the Keyboard/Arduino buttons, Linux processor on the Lab computer and the Audio component on the board. The game has multiple level, each one lasting for 60 seconds. The score from one level is carried forward to the next. The time counting down is displayed on the screen. The screen also shows the level and score. 3 Moles pop out of the ground at random and the purpose is to press the right button on the keyboard so as to make the hammer move to the position of the popping mole and strike it in time before it goes back inside the hole. Every successful hit is indicated by a 'smack' sign on the screen and the counter of the score is incremented by 1. For a failed hit or no hit, there is no action and score remains the same. As mentioned above, the goal is to hit many moles as possible with the hammer and beat your friend's score.

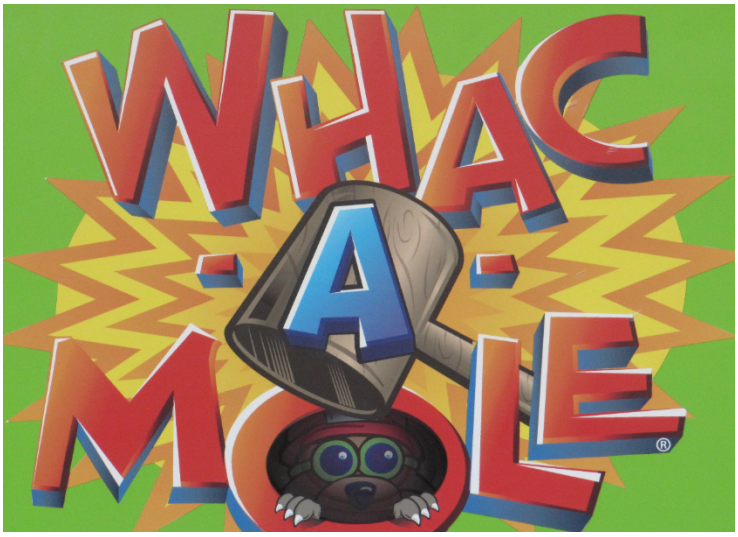

Figure 1.1

2. System Block Diagram

The highest level in our design will be based on the Processor System Block Diagram(Fig2.1). Most Processors are based on this particular system block.

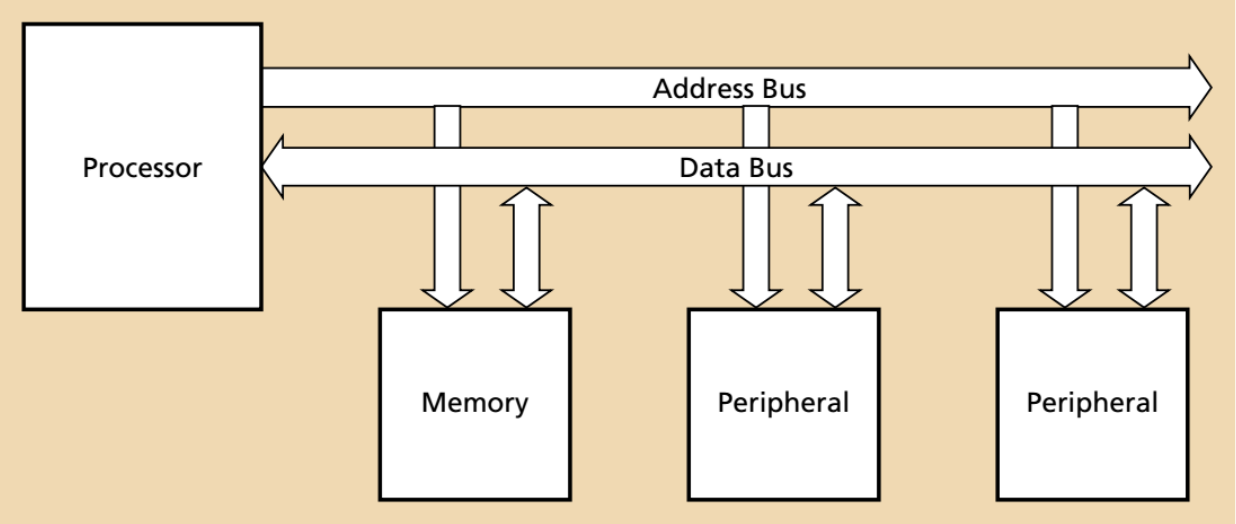

Figure 2.1 : Processor System Block Diagram

Below we outline in detail the blocks our design includes:

- **SRAM Memory**: The memory will be provided by Linux so we do not need to implement it like we did in Lab1.
- **SRAM Controller**: The memory controller is also included in Linux.
- **Linux Processor**: The processor is provided by the Linux environment in the Lab computer.
- **Avalon Bus**: SoCKit board includes the Altera Avalon bus which we used as Data and Address bus.
- **VGA Display**: The monitor for our project as a peripheral.
- **VGA Display Controller** : We created a controller for the VGA Display on the screen.
- **Audio Output and Controller**: Required audio output and controller for the different sounds of the game.

All the components and the way they are connected are shown in Fig2.2.

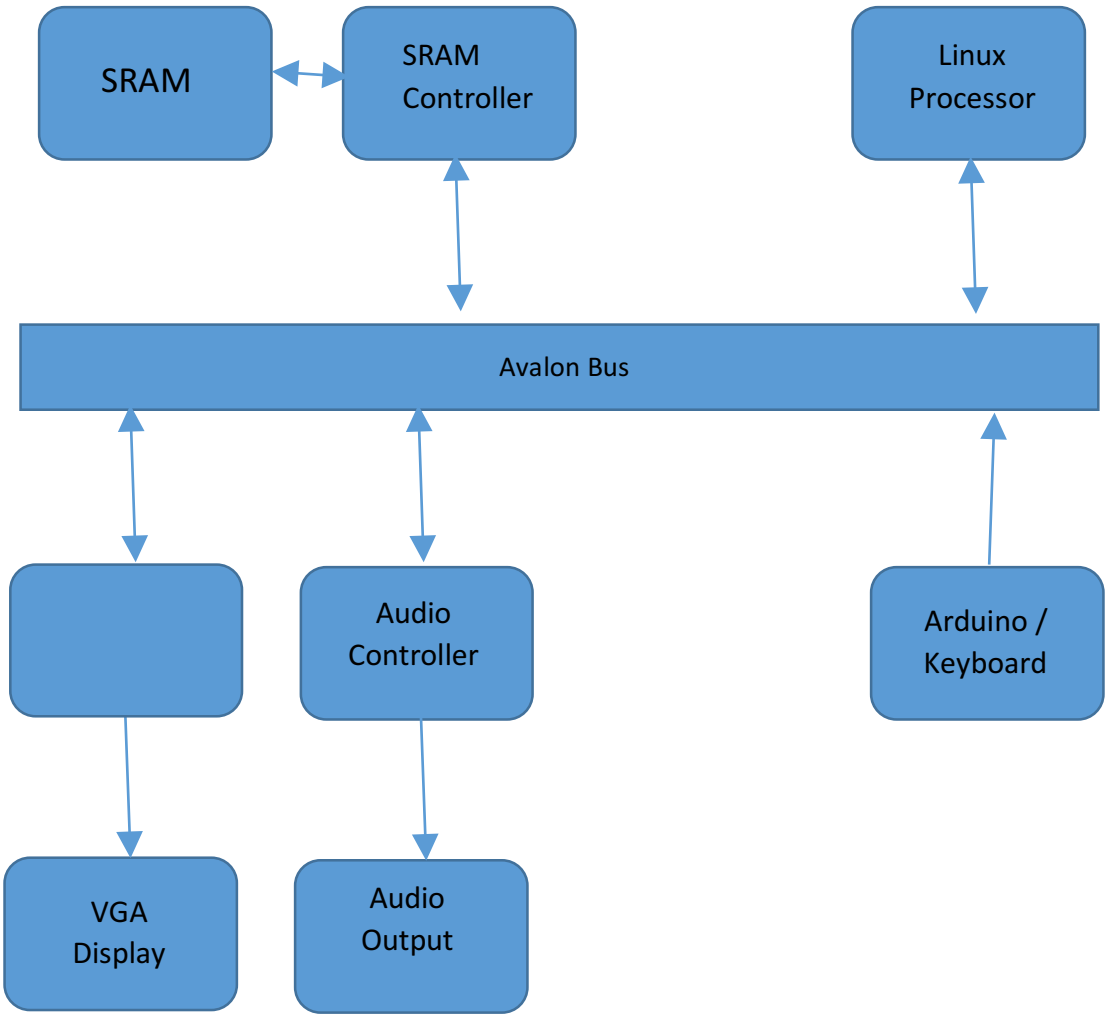

Figure 2.2: Project Block Diagram

3. Hardware Components

Processor & SRAM

VGA controller reads the variable that represents the position and shape of the hammer that is used to hit the moles. The audio controller reads the variable that represents the current playing audio.

Variables used that are stored in registers.

SRAM

This is the memory of the whole system. It is hosted on the Linux environment. It is the intermedia between the processor and the peripherals. SRAM stores the status of the game and all the shapes that appear on the screen. Audio data is also stored in the SRAM.

The Processor

This is the Linux processor, used to either read or write data to the SRAM via the Avalon Bus. The hardware and software commands are carried out via the processor.

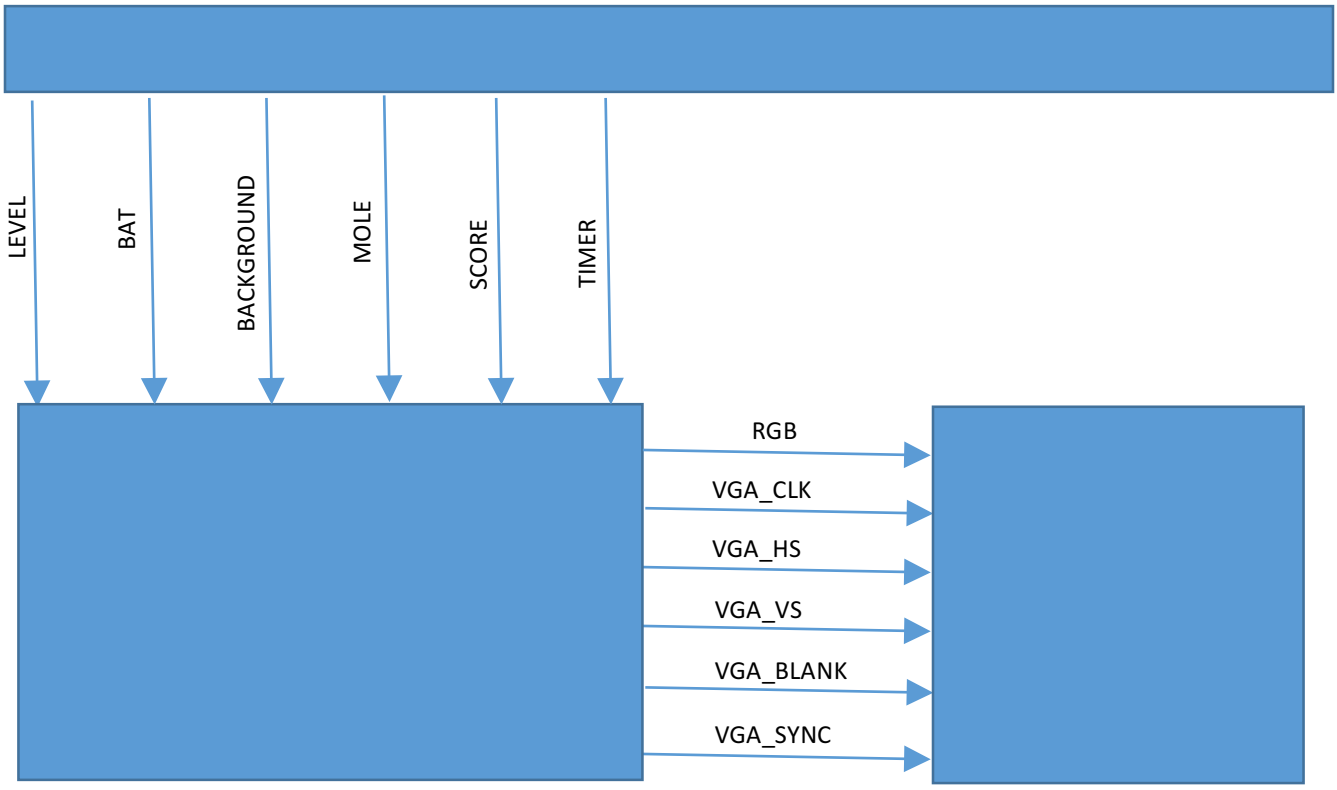

Figure 3.1: VGA Display Controller Block with ports

Audio Controller

The Audio:

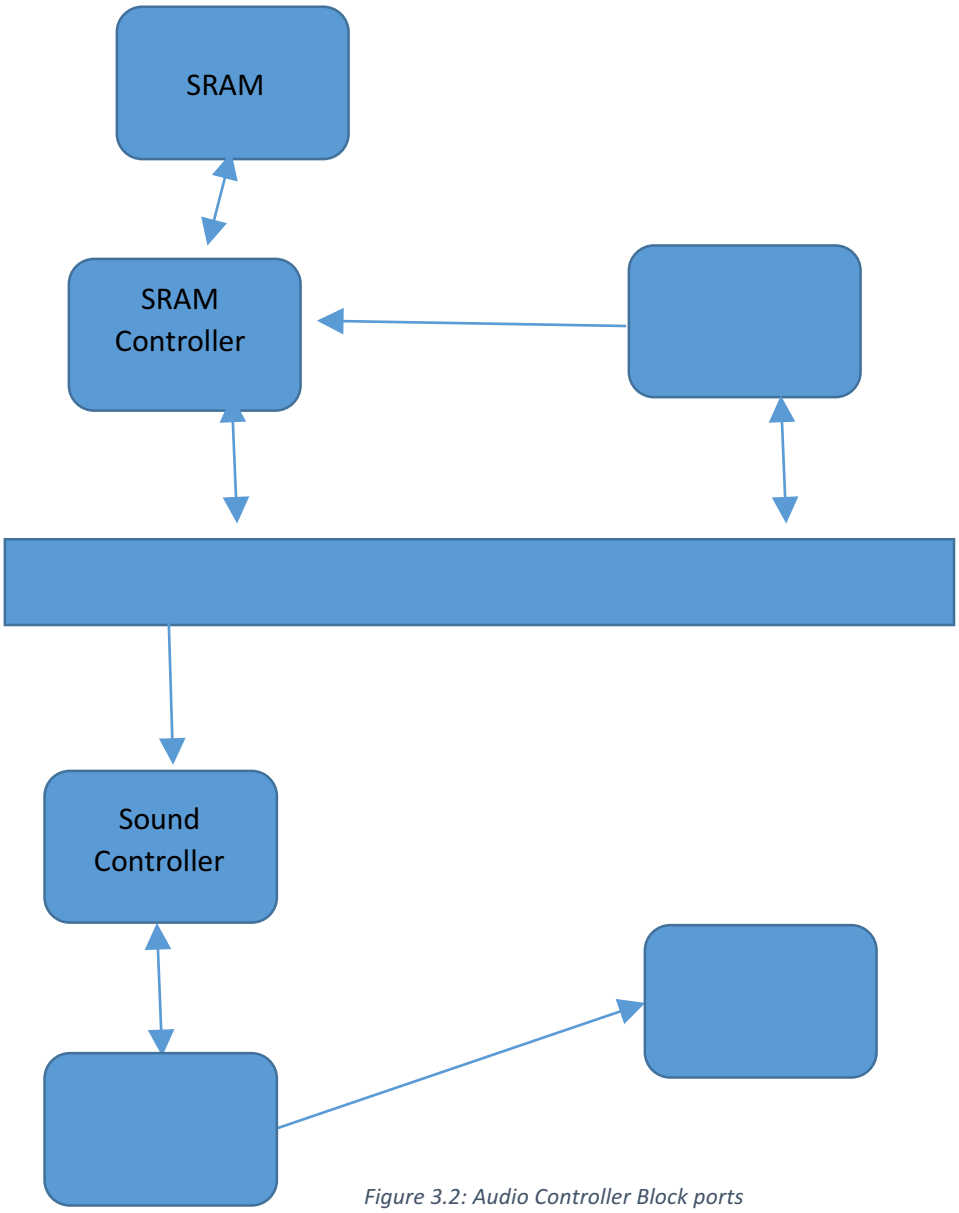

Figure 3.2: Audio Controller Block ports

- The processor controls SRAM to transfer audio files to audio peripherals.
- Audio Codec is the audio function chip on the SoCKit board.
- The Sound Hardware consists of the SystemVerilog code that is used to control the chip by providing it clocks and transferring audio data into it.
- The Sound controller maps the audio part into the bus-memory-processor system.
- The processor receives the C code that decides when to play what audio.

Memory

The graphics of our project were based on the VGA_LED module from lab3 and its submodule VGA-LED_Emulator module. The former receives data through the writedata 8-bit input.The incoming data is then stored into 8 8-bit registers named hex0-hex7. Each data is stored in a different register via the address[3:0] input. We used 6 out of eight registers for our project(hex0-hex5).

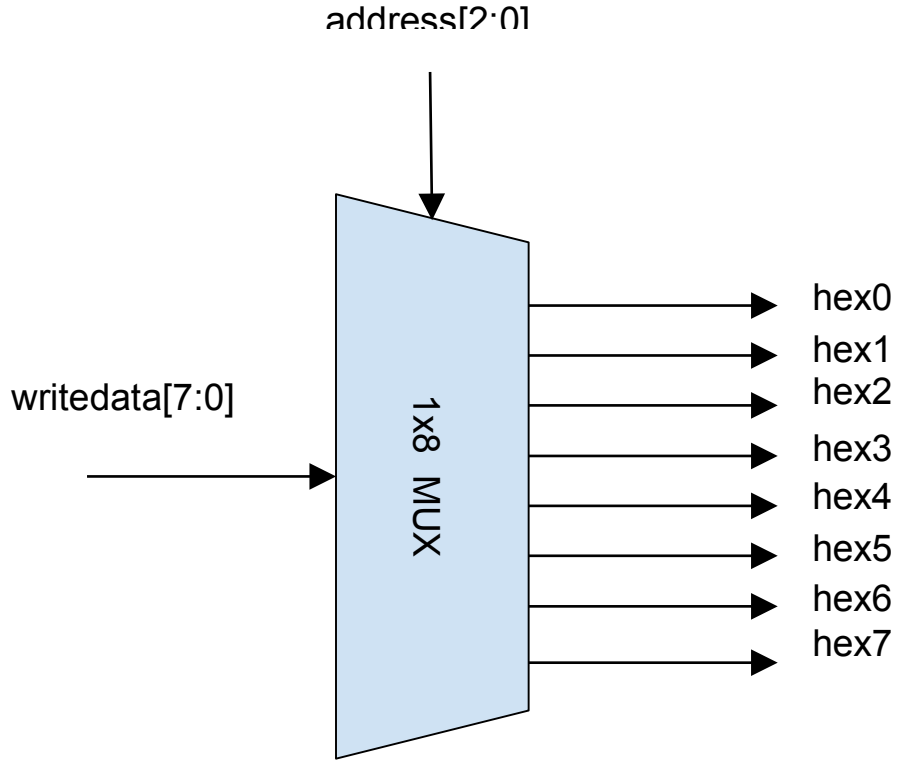

Figure 3.3: Registers

We present the mapping of our registers to the game entities. It is basically the hardware software interface.

hex0[7:0]	7	6	5	4	3	2	1	0
Usage:	Mole 2 Position				Mole 1 Position			
hex1[7:0]	7	6	5	4	3	2	1	0
Usage:	Hammer Position				Mole 3 Position			
hex2[7:0]	7	6	5	4	3	2	1	0
Usage:	Sound 8	Sound 7	Sound 6	Sound 5	Sound 4	Sound 3	Sound 2	Sound 1
hex3[7:0]	7	6	5	4	3	2	1	0
Usage:	SCORE							
hex4[7:0]	7	6	5	4	3	2	1	0
Usage:	TIME							
hex5[7:0]	7	6	5	4	3	2	1	0
Usage:	LEVEL							

Figure 3.4: Hardware Software Interface

These registers are fed into the input of the VGA_LED_Emulator submodule.From there, the raster generates 60fps to the monitor output with the hcount being the horizontal counter and the vcount the vertical one. Flip-flops triggered with the 25MHz frequency of the VGA_CLK are used for this purpose.

Graphics

With that in mind, we generate the graphics in inverse order of appearence and importance. First we generate the sky and the grass which is the basic background. This is done by painting the screen light blue and green with VGA_R, VGA_G, VGA_B. Once this is done, we generate the 9 moleholes.Each mole is instantiated as a black elipse.The elipse equation is shown below:

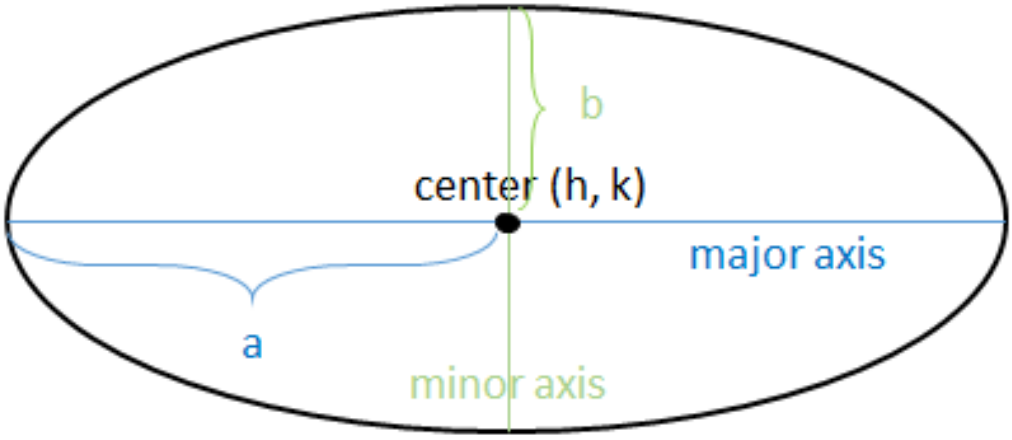

Figure 3.5: Ellipse for holes

All the rest of the graphics are created through sprites. Each entity can be constructed by one or more sprites. We used a memory map which is displayed on the screen at a certain coordinate(1st mux) and for a certain pixel area, usually the 1-valued bits(2nd mux). The mole sprite which included 7 different sprites was the most challenging one.In this case where more than 1 sprits are used, they are placed on top of each other in order of appearence.

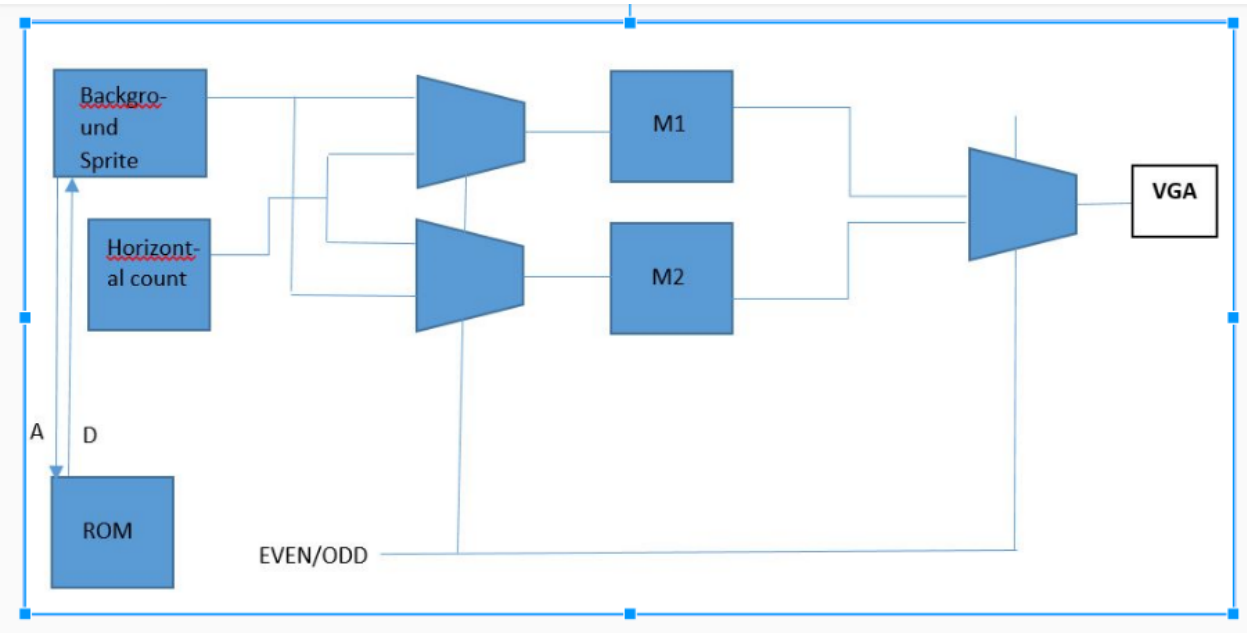

Figure 3.6: Sprite Graphics

To decide the position of the mole, we use 4 bits since we have 9 holes and thus 9 cases for the mole to pop plus 1 more which represents the absence of the mole, so a total of 10.The most demanding part for the mole though was the pop up and pop down.

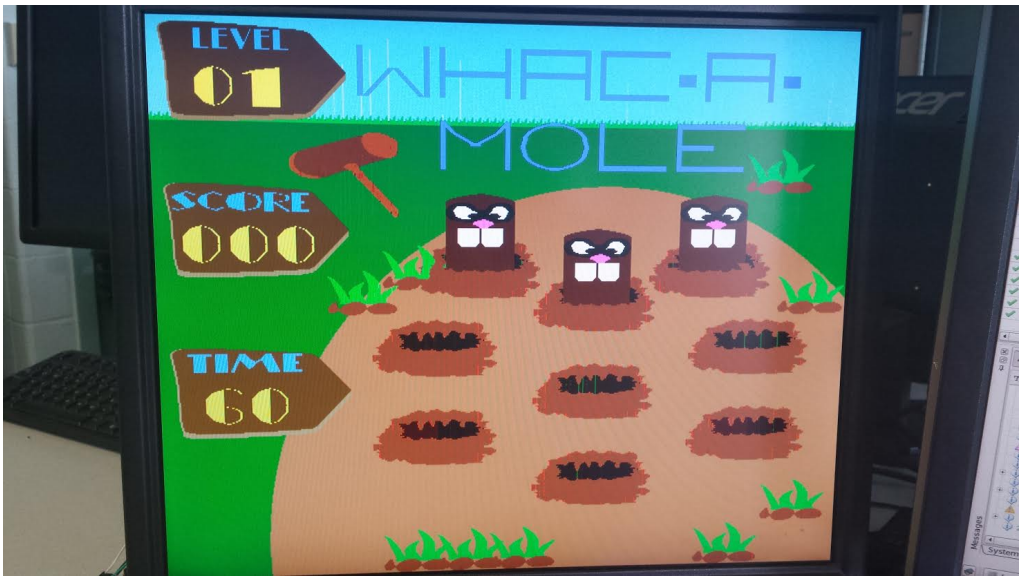

Figure 3.7: Screenshot of the Game

Mole pop

What we did, is to take advantage of the 60fps raster and use a flip flop triggered at the end of the field so 60Hz frequency. When the flip-flop is triggered , we increment a counter from 0 until the height of the mole. When it reaches that height it is hardwired to it and does not increment any more.The trick is to display the mole up until that counter. To instantiate the pop down, we follow the same procedure inversely. We decrement the same counter until it reaches 0 where it is hardwired.Again we display the mole up until that counter.

In order to make our lives easier, we decided after a mole appearence, to send that mole to the 0 case(no mole) which means not in hole.So a sequence for a mole pop would be:

0 -> 5 -> 0 -> 3 -> 0 -> 9 ->0

Level-Time-score

To display the level,the time and the score, we used 10 sprites for the numbers,multipliers and dividers to extract the decimal parts from the register and also MUX's to output the respective sprite for each number.

Hammer

The hammer is based on two sprites on top of each other. It's basel location is always at the top left of the screen. Whenever we chose to strike at a hole, it is rotated 90 degrees counter clockwise and displayed in the hole. If a strike occurs, a yellow flash sprite is being displayed.

The sequence of appearence is of paramount importance for all the above.

Audio

We instantiated an audio_codec module to output the sound effects stored in the FPGA ROM. The codec communicates with the Avalon Bus via I2C protocol.We will also instantiate i2c_controller module and i2c_av_config module from the sockit website.

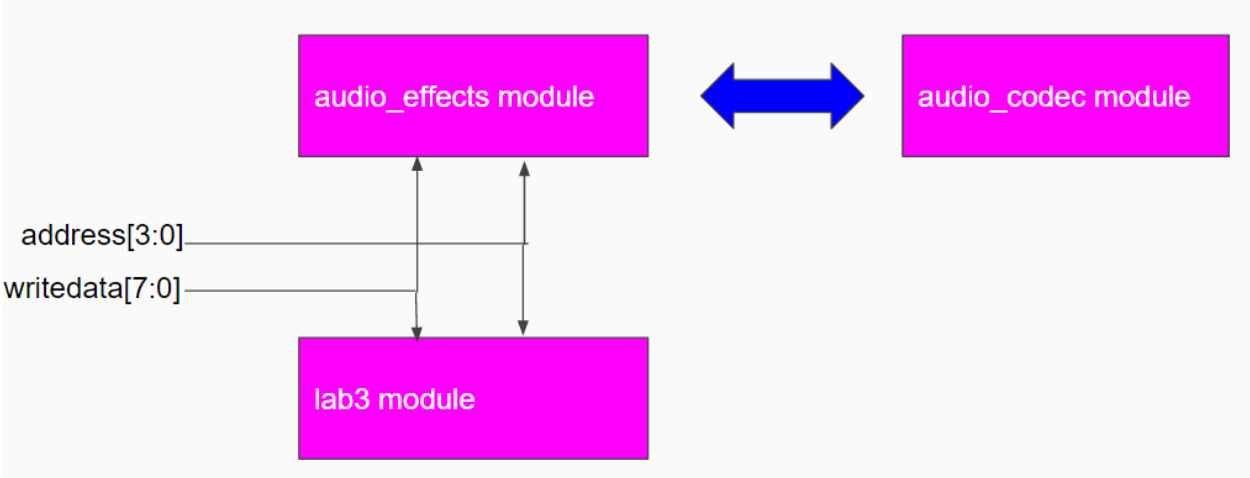

Figure 3.8: Audio Codec Interfacing

Inside audio_top, there are four parts in our design, including audio_codec module, i2c_av_config module, audio_effects module and a PLL for sample rate clock generation. The aduio_codec defines parameters of SSM2603 on board, specifically the master clock MCLK, bit clock BCLK, RECLRC and the left clock and right clock for two stereo channels RECLRC and PBLRC. These clock signals are related with the clock sample rate respectively. The sample rate clock is 44.1kHz which generated by PLL using mega wizard in Quartus. I2c_av_config drives the i2c_controller inside it. I2c_controller specifically define the communication path between FPGA as a master and audio Codec as a slave based on I2C protocol. We modify the transmitted signals as 7-bit address, 1-bit read/write enable and 8 bit data. The driver specify the several requirements and we basically increase the left and right sound volume into 6dB.

In the audio_effect module, we specify our sounds. All sounds are stored in ROMs and used in this module. We generated a background sound for a design which was controlled from the hardware. The .mif file for the audio was generated with the help of MATLAB processing and this was read into the ROM.

To control the sound, we need to draw a wire from the lab3 module input of hex2 and address and connect them to the audio_effects module as shown in the figure above.Of course we would need to add more waveform rom files and some combinational logic comprised mainly of mux's to the audio_effects module.

4. Software

The software part was comprised of the main hello.c program, the vga_led.c device driver for VGA_LED module from lab3 and the usbkeyboard.c program from lab2.

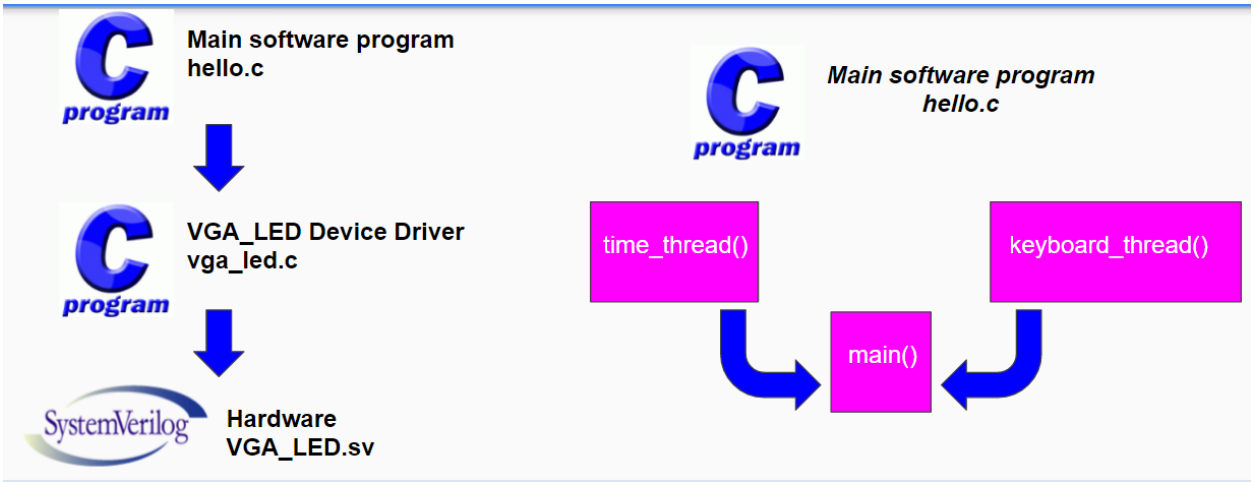

Figure 4.1: Software Interface

hello.c

The software overview for controlling our hardware is shown above. The hello.c is taken from lab3 and extended. It includes the main() function which is the skeleton and the time_thread and keyboard_thread running on the background. Also it uses the write_segments function to send the data to the device driver.

The time thread, instantiates a timer for our game. It counts down from 60 seconds until 0 when we move to the next level. Every operation like the popping of the mole, the level duration etc is synchronised with that particular timer thread.

The keyboard thread opens the usb input and expects an input from the keyboard or the arduino.It mainly uses the libusb_interrupt_transfer function located in the libusb-1.0.0 library. This function pauses the program until the user pushes a button. This is why it should be put into the background as a thread. These two threads need to be handled with care in order to avoid multithreading issues. The main concern is that thread used variables should not be manipulated in the main body or by other threads.

Figure 4.2: Multithreading

The message[8] variable includes the data for our 8 registers.message[0] goes to hex0, message[1] to hex1 etc.

The main function starts by opening the usb interface and checking for the keyboard. Once it is found it opens the 2 threads. Afterwards it enters an infinite loop. In that loop, the essential stuff takes place in the sequence below:

i. Assign timer to message[4]
ii. Check if the time is over to increment the level and pause for 1 second
iii. Change the position of each mole every 1s and do not let 2 moles appear in the same hole
iv. Use boolean (shift,or,and operations) logic to assign the mole and hammer positions to message[0] and message[1]
v. Check the hammer position
vi. For every case of position, assign the hammer position to message[1] and if there is a strike, increment the score by 1, send that mole to 0 (in no hole), assign the score to message[3] and us the write_message function to send the data to the driver.

We did not manage our time properly to implement the soundwaves. The way to do that would be for every one of the above cases to assert a different message[2] bit for a different

sound. That would be the signal to start playing that specific sound. In the case of a strike for example (number 6) we would assert message[2] to 1 and deassert it in the next cycle.

Arduino Leonardo:

The purpose of the game is to hit the popping mole on the screen.

While this function can be achieved using the keys on the keyboard, it is far more elegant to have real push buttons for this task.

The Arduino Leonardo allows us to interface push buttons on a breadboard with the Cyclone V board.

The Arduino can be programmed to behave as a keyboard based on the button pressing.

Eventually, we can have a push button system hooked up to the Cyclone V board via the Arduino and push the appropriate buttons to Whac the corresponding Mole.

Arduino Working:

Unfortunately, we could not manage to have the push buttons on the Arduino to function correctly in time for the presentation and final submission. We implemented the Arduino before the pthreads that allow multiple programs to run simultaneously however the functioning seemed to deteriorate when the pthreads were used. We believe the problem lies in the package being sent by the Arduino to the board is not recognized as a keyboard and even though we modified usbkeyboard.c to solve this problem, we believe that a new driver must be designed for the Arduino to function as a keyboard as the ioctl function is not giving the right result with the Arduino.

5. Future Work and Conclusion

Incomplete goals

Even after initial success of implementing the logic to use Arduino Leonardo as keyboard, we were unable to incorporate it in the final design because of addition of threads.

The sound waves could be controlled only from the hardware. We can extend it and control it from the software resulting in different audio effects for different activities in

the game. Currently we were able to implement the background audio only.

Lessons learned

1.It is better to split the design in small modules.

1.Approach issues in parallel. Not in series. The ones in the end might require more time than planned. By dealing with them for a certain amount of time in the beginning you can do more accurate predictions about the workload.

Future work

1. Insert software logic for increasing mole speed for every next level.

2. Add more sophisticated graphics (mole strike, wind moving, grass, birds, cows in the background)

3. Add more mole stripes.

4. Incorporate main menu and player name interface.

5. Control the audio from software.

Conclusion

We are able to successfully complete a working demonstration of the Whac-A-Mole game. The final version of the game allows the user to Whac the Moles using the keyboard buttons and records the score simultaneously for successful hits. The time counts down from 60 seconds for each level and the final score is displayed on the screen.

6. Code

Arduino Code:

```
#include <Keyboard.h>

void setup() {
  // make pin 2 an input and turn on the
```

```
  // pullup resistor so it goes high unless
  // connected to ground:
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
  pinMode(7, INPUT_PULLUP);
  pinMode(8, INPUT_PULLUP);
  pinMode(9, INPUT_PULLUP);
  pinMode(10, INPUT_PULLUP);
  Keyboard.begin();
}

void loop() {
  //if the button is pressed
  if(digitalRead(2)==LOW){
    Keyboard.write(0x1e);
    delay(200);
  }
  if(digitalRead(3)==LOW){
    Keyboard.write(0x1f);
    delay(200);
  }
  if(digitalRead(4)==LOW){
    Keyboard.write(0x20);
    delay(200);
```

```
  }
  if(digitalRead(5)==LOW){
    Keyboard.write(0x21);
    delay(200);
  }
  if(digitalRead(6)==LOW){
    Keyboard.write(0x22);
    delay(200);
  }
  if(digitalRead(7)==LOW){
    Keyboard.write(0x23);
    delay(200);
  }
  if(digitalRead(8)==LOW){
    Keyboard.write(0x24);
    delay(200);
  }
  if(digitalRead(9)==LOW){
    Keyboard.write(0x25);
    delay(200);
  }
  if(digitalRead(10)==LOW){
    Keyboard.write(0x26);
    delay(200);
  }
}
```

audio_codec.sv

```
module audio_codec (
   input  clk,
   input  reset,
   output [1:0]  sample_end,
   output [1:0]  sample_req,
   input  [15:0] audio_output,
   output [15:0] audio_input,
   // 1 - left, 0 - right
   input  [1:0] channel_sel,

   output AUD_ADCLRCK,
   input AUD_ADCDAT,
   output AUD_DACLRCK,
   output AUD_DACDAT,
   output AUD_BCLK
);

reg [7:0] lrck_divider;
reg [1:0] bclk_divider;

reg [15:0] shift_out;
reg [15:0] shift_temp;
reg [15:0] shift_in;

wire lrck = !lrck_divider[7];
```

```
assign AUD_ADCLRCK = lrck;
assign AUD_DACLRCK = lrck;
assign AUD_BCLK = bclk_divider[1];
assign AUD_DACDAT = shift_out[15];

always @(posedge clk) begin
   if (reset) begin
      lrck_divider <= 8'hff;
      bclk_divider <= 2'b11;
   end else begin
      lrck_divider <= lrck_divider + 1'b1;
      bclk_divider <= bclk_divider + 1'b1;
   end
end

assign sample_end[1] = (lrck_divider == 8'h40);
assign sample_end[0] = (lrck_divider == 8'hc0);
assign audio_input = shift_in;
assign sample_req[1] = (lrck_divider == 8'hfe);
assign sample_req[0] = (lrck_divider == 8'h7e);

wire clr_lrck = (lrck_divider == 8'h7f);
wire set_lrck = (lrck_divider == 8'hff);
// high right after bclk is set
wire set_bclk = (bclk_divider == 2'b10 && !lrck_divider[6]);
// high right before bclk is cleared
wire clr_bclk = (bclk_divider == 2'b11 && !lrck_divider[6]);
```

```
always @(posedge clk) begin
    if (reset) begin
        shift_out <= 16'h0;
        shift_in <= 16'h0;
        shift_in <= 16'h0;
    end else if (set_lrck || clr_lrck) begin
        // check if current channel is selected
        if (channel_sel[set_lrck]) begin
            shift_out <= audio_output;
            shift_temp <= audio_output;
            shift_in <= 16'h0;
        // repeat the sample from the other channel if not
        end else shift_out <= shift_temp;
    end else if (set_bclk == 1) begin
        // only read in if channel is selected
        if (channel_sel[lrck])
            shift_in <= {shift_in[14:0], AUD_ADCDAT};
    end else if (clr_bclk == 1) begin
        shift_out <= {shift_out[14:0], 1'b0};
    end
end


endmodule
```

audio_effects.sv

```
module audio_effects (
```

```
   input  clk,
   input  sample_end,
   input  sample_req,
   output [15:0] audio_output,
   input  [15:0] audio_input,
   input  [3:0]  control,
          input [15:0] sound_new,
          output [14:0] addr,
          input [7:0] writedata,
          input [2:0] address
);

//reg [15:0] romdata [0:99];
//reg [6:0]  index = 7'd0;
//index_addr = 15'd0;
reg [15:0] last_sample;
reg [15:0] dat;
reg [14:0] index_addr = 15'd0;
//assign index_addr = addr;
always @(posedge clk)
begin
addr <= index_addr;
end

assign audio_output = dat;

parameter SINE     = 0;
```

```
parameter FEEDBACK = 1;

/*initial begin
    romdata[0] = 16'h0000;
    romdata[1] = 16'h0805;
    romdata[2] = 16'h1002;
    romdata[3] = 16'h17ee;
    romdata[4] = 16'h1fc3;
    romdata[5] = 16'h2777;
    romdata[6] = 16'h2f04;
    romdata[7] = 16'h3662;
    romdata[8] = 16'h3d89;
    romdata[9] = 16'h4472;
    romdata[10] = 16'h4b16;
    romdata[11] = 16'h516f;
    romdata[12] = 16'h5776;
    romdata[13] = 16'h5d25;
    romdata[14] = 16'h6276;
    romdata[15] = 16'h6764;
    romdata[16] = 16'h6bea;
    romdata[17] = 16'h7004;
    romdata[18] = 16'h73ad;
    romdata[19] = 16'h76e1;
    romdata[20] = 16'h799e;
    romdata[21] = 16'h7be1;
    romdata[22] = 16'h7da7;
    romdata[23] = 16'h7eef;
```

```
romdata[24] = 16'h7fb7;
romdata[25] = 16'h7fff;
romdata[26] = 16'h7fc6;
romdata[27] = 16'h7f0c;
romdata[28] = 16'h7dd3;
romdata[29] = 16'h7c1b;
romdata[30] = 16'h79e6;
romdata[31] = 16'h7737;
romdata[32] = 16'h7410;
romdata[33] = 16'h7074;
romdata[34] = 16'h6c67;
romdata[35] = 16'h67ed;
romdata[36] = 16'h630a;
romdata[37] = 16'h5dc4;
romdata[38] = 16'h5820;
romdata[39] = 16'h5222;
romdata[40] = 16'h4bd3;
romdata[41] = 16'h4537;
romdata[42] = 16'h3e55;
romdata[43] = 16'h3735;
romdata[44] = 16'h2fdd;
romdata[45] = 16'h2855;
romdata[46] = 16'h20a5;
romdata[47] = 16'h18d3;
romdata[48] = 16'h10e9;
romdata[49] = 16'h08ee;
romdata[50] = 16'h00e9;
```

```
romdata[51] = 16'hf8e4;
romdata[52] = 16'hf0e6;
romdata[53] = 16'he8f7;
romdata[54] = 16'he120;
romdata[55] = 16'hd967;
romdata[56] = 16'hd1d5;
romdata[57] = 16'hca72;
romdata[58] = 16'hc344;
romdata[59] = 16'hbc54;
romdata[60] = 16'hb5a7;
romdata[61] = 16'haf46;
romdata[62] = 16'ha935;
romdata[63] = 16'ha37c;
romdata[64] = 16'h9e20;
romdata[65] = 16'h9926;
romdata[66] = 16'h9494;
romdata[67] = 16'h906e;
romdata[68] = 16'h8cb8;
romdata[69] = 16'h8976;
romdata[70] = 16'h86ab;
romdata[71] = 16'h845a;
romdata[72] = 16'h8286;
romdata[73] = 16'h8130;
romdata[74] = 16'h8059;
romdata[75] = 16'h8003;
romdata[76] = 16'h802d;
romdata[77] = 16'h80d8;
```

```
    romdata[78] = 16'h8203;
    romdata[79] = 16'h83ad;
    romdata[80] = 16'h85d3;
    romdata[81] = 16'h8875;
    romdata[82] = 16'h8b8f;
    romdata[83] = 16'h8f1d;
    romdata[84] = 16'h931e;
    romdata[85] = 16'h978c;
    romdata[86] = 16'h9c63;
    romdata[87] = 16'ha19e;
    romdata[88] = 16'ha738;
    romdata[89] = 16'had2b;
    romdata[90] = 16'hb372;
    romdata[91] = 16'hba05;
    romdata[92] = 16'hc0df;
    romdata[93] = 16'hc7f9;
    romdata[94] = 16'hcf4b;
    romdata[95] = 16'hd6ce;
    romdata[96] = 16'hde7a;
    romdata[97] = 16'he648;
    romdata[98] = 16'hee30;
    romdata[99] = 16'hf629;
end
*/
always @(posedge clk) begin
    if (sample_end) begin
        last_sample <= audio_input;
```

```
    end

    if (sample_req) begin
        if (control[FEEDBACK])
            dat <= last_sample;
                                //sound_new <= last_sample;
        else if ((control[SINE])& (address==2) & (writedata==1)) begin
            dat <= sound_new;
            if (index_addr == 15'd22049)
                index_addr <= 15'd0;
            else
                index_addr <= index_addr + 1'b1;
        end else
            dat <= 16'd0;
    end
end

endmodule
```

clock_pll.v

```
// megafunction wizard: %Altera PLL v13.1%
// GENERATION: XML
// clock_pll.v

// Generated using ACDS version 13.1.1 166 at 2016.05.11.22:49:45

`timescale 1 ps / 1 ps
module clock_pll (
```

```
            input  wire  refclk,   //  refclk.clk
            input  wire  rst,     //   reset.reset
            output wire  outclk_0, // outclk0.clk
            output wire  outclk_1  // outclk1.clk
        );

        clock_pll_0002 clock_pll_inst (
            .refclk   (refclk),   //  refclk.clk
            .rst      (rst),      //   reset.reset
            .outclk_0 (outclk_0), // outclk0.clk
            .outclk_1 (outclk_1), // outclk1.clk
            .locked   ()        // (terminated)
        );

endmodule
```

i2c_av_config.sv

```
module i2c_av_config (
   input clk,
   input reset,

   output i2c_sclk,
   inout  i2c_sdat,

   output [3:0] status
);

reg [23:0] i2c_data;
reg [15:0] lut_data;
```

```
reg [3:0]  lut_index = 4'd0;

parameter LAST_INDEX = 4'ha;

reg  i2c_start = 1'b0;
wire i2c_done;
wire i2c_ack;

i2c_controller control (
   .clk (clk),
   .i2c_sclk (i2c_sclk),
   .i2c_sdat (i2c_sdat),
   .i2c_data (i2c_data),
   .start (i2c_start),
   .done (i2c_done),
   .ack (i2c_ack)
);

always @(*) begin
   case (lut_index)
      4'h0: lut_data <= 16'h0c10; // power on everything except out
      4'h1: lut_data <= 16'h0017; // left input
      4'h2: lut_data <= 16'h0217; // right input
      4'h3: lut_data <= 16'h0479; // left output
      4'h4: lut_data <= 16'h0679; // right output
      4'h5: lut_data <= 16'h08d4; // analog path
      4'h6: lut_data <= 16'h0a04; // digital path
      4'h7: lut_data <= 16'h0e01; // digital IF
      4'h8: lut_data <= 16'h1020; // sampling rate
```

```
        4'h9: lut_data <= 16'h0c00; // power on everything
        4'ha: lut_data <= 16'h1201; // activate
        default: lut_data <= 16'h0000;
    endcase
end

reg [1:0] control_state = 2'b00;

assign status = lut_index;

always @(posedge clk) begin
    if (reset) begin
        lut_index <= 4'd0;
        i2c_start <= 1'b0;
        control_state <= 2'b00;
    end else begin
        case (control_state)
            2'b00: begin
                i2c_start <= 1'b1;
                i2c_data <= {8'h34, lut_data};
                control_state <= 2'b01;
            end
            2'b01: begin
                i2c_start <= 1'b0;
                control_state <= 2'b10;
            end
            2'b10: if (i2c_done) begin
                if (i2c_ack) begin
                    if (lut_index == LAST_INDEX)
```

```
                control_state <= 2'b11;
            else begin
                lut_index <= lut_index + 1'b1;
                control_state <= 2'b00;
            end
        end else
            control_state <= 2'b00;
    end
  endcase
 end
end

endmodule
```

i2c_controller.sv

```
module i2c_controller (
  input  clk,

  output i2c_sclk,
  inout  i2c_sdat,

  input  start,
  output done,
  output ack,

  input [23:0] i2c_data
);

reg [23:0] data;
```

```
reg [4:0] stage;
reg [6:0] sclk_divider;
reg clock_en = 1'b0;

// don't toggle the clock unless we're sending data
// clock will also be kept high when sending START and STOP symbols
assign i2c_sclk = (!clock_en) || sclk_divider[6];
wire midlow = (sclk_divider == 7'h1f);

reg sdat = 1'b1;
// rely on pull-up resistor to set SDAT high
assign i2c_sdat = (sdat) ? 1'bz : 1'b0;

reg [2:0] acks;

parameter LAST_STAGE = 5'd29;

assign ack = (acks == 3'b000);
assign done = (stage == LAST_STAGE);

always @(posedge clk) begin
   if (start) begin
      sclk_divider <= 7'd0;
      stage <= 5'd0;
      clock_en = 1'b0;
      sdat <= 1'b1;
      acks <= 3'b111;
      data <= i2c_data;
```

```
end else begin
   if (sclk_divider == 7'd127) begin
      sclk_divider <= 7'd0;

      if (stage != LAST_STAGE)
         stage <= stage + 1'b1;

      case (stage)
         // after start
         5'd0:  clock_en <= 1'b1;
         // receive acks
         5'd9:  acks[0] <= i2c_sdat;
         5'd18: acks[1] <= i2c_sdat;
         5'd27: acks[2] <= i2c_sdat;
         // before stop
         5'd28: clock_en <= 1'b0;
      endcase
   end else
      sclk_divider <= sclk_divider + 1'b1;

   if (midlow) begin
      case (stage)
         // start
         5'd0:  sdat <= 1'b0;
         // byte 1
         5'd1:  sdat <= data[23];
         5'd2:  sdat <= data[22];
         5'd3:  sdat <= data[21];
         5'd4:  sdat <= data[20];
```

```
5'd5:  sdat <= data[19];
5'd6:  sdat <= data[18];
5'd7:  sdat <= data[17];
5'd8:  sdat <= data[16];
// ack 1
5'd9:  sdat <= 1'b1;
// byte 2
5'd10: sdat <= data[15];
5'd11: sdat <= data[14];
5'd12: sdat <= data[13];
5'd13: sdat <= data[12];
5'd14: sdat <= data[11];
5'd15: sdat <= data[10];
5'd16: sdat <= data[9];
5'd17: sdat <= data[8];
// ack 2
5'd18: sdat <= 1'b1;
// byte 3
5'd19: sdat <= data[7];
5'd20: sdat <= data[6];
5'd21: sdat <= data[5];
5'd22: sdat <= data[4];
5'd23: sdat <= data[3];
5'd24: sdat <= data[2];
5'd25: sdat <= data[1];
5'd26: sdat <= data[0];
// ack 3
5'd27: sdat <= 1'b1;
// stop
```

```
            5'd28: sdat <= 1'b0;
            5'd29: sdat <= 1'b1;
          endcase
       end
    end
end

endmodule
```

SoCKit_top.v

```
//`define ENABLE_DDR3
//`define ENABLE_HPS
//`define ENABLE_HSMC_XCVR

module SoCKit_Top(

                        ///////////AUD/////////////
                        AUD_ADCDAT,
                        AUD_ADCLRCK,
                        AUD_BCLK,
                        AUD_DACDAT,
                        AUD_DACLRCK,
                        AUD_I2C_SCLK,
                        AUD_I2C_SDAT,
                        AUD_MUTE,
                        AUD_XCK,

`ifdef ENABLE_DDR3
```

```
                ////////DDR3////////
                DDR3_A,
                DDR3_BA,
                DDR3_CAS_n,
                DDR3_CKE,
                DDR3_CK_n,
                DDR3_CK_p,
                DDR3_CS_n,
                DDR3_DM,
                DDR3_DQ,
                DDR3_DQS_n,
                DDR3_DQS_p,
                DDR3_ODT,
                DDR3_RAS_n,
                DDR3_RESET_n,
                DDR3_RZQ,
                DDR3_WE_n,
`endif /*ENABLE_DDR3*/

                ////////FAN////////
                FAN_CTRL,

`ifdef ENABLE_HPS
                ////////HPS////////
                HPS_CLOCK_25,
                HPS_CLOCK_50,
                HPS_CONV_USB_n,
                HPS_DDR3_A,
                HPS_DDR3_BA,
```

HPS_DDR3_CAS_n,

HPS_DDR3_CKE,

HPS_DDR3_CK_n,

HPS_DDR3_CK_p,

HPS_DDR3_CS_n,

HPS_DDR3_DM,

HPS_DDR3_DQ,

HPS_DDR3_DQS_n,

HPS_DDR3_DQS_p,

HPS_DDR3_ODT,

HPS_DDR3_RAS_n,

HPS_DDR3_RESET_n,

HPS_DDR3_RZQ,

HPS_DDR3_WE_n,

HPS_ENET_GTX_CLK,

HPS_ENET_INT_n,

HPS_ENET_MDC,

HPS_ENET_MDIO,

HPS_ENET_RESET_n,

HPS_ENET_RX_CLK,

HPS_ENET_RX_DATA,

HPS_ENET_RX_DV,

HPS_ENET_TX_DATA,

HPS_ENET_TX_EN,

HPS_FLASH_DATA,

HPS_FLASH_DCLK,

HPS_FLASH_NCSO,

HPS_GSENSOR_INT,

HPS_I2C_CLK,

```
                HPS_I2C_SDA,
                HPS_KEY,
                HPS_LCM_D_C,
                HPS_LCM_RST_N,
                HPS_LCM_SPIM_CLK,
                HPS_LCM_SPIM_MISO,
                HPS_LCM_SPIM_MOSI,
                HPS_LCM_SPIM_SS,
                HPS_LED,
                HPS_LTC_GPIO,
                HPS_RESET_n,
                HPS_SD_CLK,
                HPS_SD_CMD,
                HPS_SD_DATA,
                HPS_SPIM_CLK,
                HPS_SPIM_MISO,
                HPS_SPIM_MOSI,
                HPS_SPIM_SS,
                HPS_SW,
                HPS_UART_RX,
                HPS_UART_TX,
                HPS_USB_CLKOUT,
                HPS_USB_DATA,
                HPS_USB_DIR,
                HPS_USB_NXT,
                HPS_USB_RESET_PHY,
                HPS_USB_STP,
                HPS_WARM_RST_n,
`endif /*ENABLE_HPS*/
```

```
                ////////HSMC////////
                HSMC_CLKIN_n,
                HSMC_CLKIN_p,
                HSMC_CLKOUT_n,
                HSMC_CLKOUT_p,
                HSMC_CLK_IN0,
                HSMC_CLK_OUT0,
                HSMC_D,

`ifdef ENABLE_HSMC_XCVR

                HSMC_GXB_RX_p,
                HSMC_GXB_TX_p,
                HSMC_REF_CLK_p,
`endif
                HSMC_RX_n,
                HSMC_RX_p,
                HSMC_SCL,
                HSMC_SDA,
                HSMC_TX_n,
                HSMC_TX_p,

                ////////IRDA////////
                IRDA_RXD,

                ////////KEY////////
                KEY,
```

```
/////////LED////////
LED,

/////////OSC////////
OSC_50_B3B,
OSC_50_B4A,
OSC_50_B5B,
OSC_50_B8A,

/////////PCIE////////
PCIE_PERST_n,
PCIE_WAKE_n,

/////////RESET////////
RESET_n,

/////////SI5338////////
SI5338_SCL,
SI5338_SDA,

/////////SW////////
SW,

/////////TEMP////////
TEMP_CS_n,
TEMP_DIN,
TEMP_DOUT,
TEMP_SCLK,
```

/////////USB////////

USB_B2_CLK,

USB_B2_DATA,

USB_EMPTY,

USB_FULL,

USB_OE_n,

USB_RD_n,

USB_RESET_n,

USB_SCL,

USB_SDA,

USB_WR_n,

/////////VGA////////

VGA_B,

VGA_BLANK_n,

VGA_CLK,

VGA_G,

VGA_HS,

VGA_R,

VGA_SYNC_n,

VGA_VS,

//////////hps/////////

memory_mem_a,

memory_mem_ba,

memory_mem_ck,

memory_mem_ck_n,

memory_mem_cke,

memory_mem_cs_n,

memory_mem_ras_n,

memory_mem_cas_n,

memory_mem_we_n,

memory_mem_reset_n,

memory_mem_dq,

memory_mem_dqs,

memory_mem_dqs_n,

memory_mem_odt,

memory_mem_dm,

memory_oct_rzqin,

hps_io_hps_io_emac1_inst_TX_CLK,

hps_io_hps_io_emac1_inst_TXD0,

hps_io_hps_io_emac1_inst_TXD1,

hps_io_hps_io_emac1_inst_TXD2,

hps_io_hps_io_emac1_inst_TXD3,

hps_io_hps_io_emac1_inst_RXD0,

hps_io_hps_io_emac1_inst_MDIO,

hps_io_hps_io_emac1_inst_MDC,

hps_io_hps_io_emac1_inst_RX_CTL,

hps_io_hps_io_emac1_inst_TX_CTL,

hps_io_hps_io_emac1_inst_RX_CLK,

hps_io_hps_io_emac1_inst_RXD1,

hps_io_hps_io_emac1_inst_RXD2,

hps_io_hps_io_emac1_inst_RXD3,

hps_io_hps_io_qspi_inst_IO0,

hps_io_hps_io_qspi_inst_IO1,

hps_io_hps_io_qspi_inst_IO2,

hps_io_hps_io_qspi_inst_IO3,

hps_io_hps_io_qspi_inst_SS0,

hps_io_hps_io_qspi_inst_CLK,

hps_io_hps_io_sdio_inst_CMD,

hps_io_hps_io_sdio_inst_D0,

hps_io_hps_io_sdio_inst_D1,

hps_io_hps_io_sdio_inst_CLK,

hps_io_hps_io_sdio_inst_D2,

hps_io_hps_io_sdio_inst_D3,

hps_io_hps_io_usb1_inst_D0,

hps_io_hps_io_usb1_inst_D1,

hps_io_hps_io_usb1_inst_D2,

hps_io_hps_io_usb1_inst_D3,

hps_io_hps_io_usb1_inst_D4,

hps_io_hps_io_usb1_inst_D5,

hps_io_hps_io_usb1_inst_D6,

hps_io_hps_io_usb1_inst_D7,

hps_io_hps_io_usb1_inst_CLK,

hps_io_hps_io_usb1_inst_STP,

hps_io_hps_io_usb1_inst_DIR,

hps_io_hps_io_usb1_inst_NXT,

hps_io_hps_io_spim0_inst_CLK,

hps_io_hps_io_spim0_inst_MOSI,

hps_io_hps_io_spim0_inst_MISO,

hps_io_hps_io_spim0_inst_SS0,

hps_io_hps_io_spim1_inst_CLK,

hps_io_hps_io_spim1_inst_MOSI,

hps_io_hps_io_spim1_inst_MISO,

hps_io_hps_io_spim1_inst_SS0,

hps_io_hps_io_uart0_inst_RX,

hps_io_hps_io_uart0_inst_TX,

hps_io_hps_io_i2c1_inst_SDA,

```
                hps_io_hps_io_i2c1_inst_SCL,
                hps_io_hps_io_gpio_inst_GPIO00
                );

//=======================================================
//  PORT declarations
//=======================================================

///////// AUD /////////
input                       AUD_ADCDAT;
inout                       AUD_ADCLRCK;
inout                       AUD_BCLK;
output                      AUD_DACDAT;
inout                       AUD_DACLRCK;
output                      AUD_I2C_SCLK;
inout                       AUD_I2C_SDAT;
output                      AUD_MUTE;
output                      AUD_XCK;

`ifdef ENABLE_DDR3
///////// DDR3 /////////
output [14:0]                   DDR3_A;
output [2:0]                    DDR3_BA;
output                      DDR3_CAS_n;
output                      DDR3_CKE;
output                      DDR3_CK_n;
output                      DDR3_CK_p;
output                      DDR3_CS_n;
output [3:0]                    DDR3_DM;
```

```
  inout [31:0]                    DDR3_DQ;
  inout [3:0]                         DDR3_DQS_n;
  inout [3:0]                         DDR3_DQS_p;
  output                  DDR3_ODT;
  output                  DDR3_RAS_n;
  output                  DDR3_RESET_n;
  input                   DDR3_RZQ;
  output                  DDR3_WE_n;
`endif /*ENABLE_DDR3*/

  ///////// FAN /////////
  output                  FAN_CTRL;

`ifdef ENABLE_HPS
  ///////// HPS /////////
  input                   HPS_CLOCK_25;
  input                   HPS_CLOCK_50;
  input                   HPS_CONV_USB_n;
  output [14:0]                   HPS_DDR3_A;
  output [2:0]                    HPS_DDR3_BA;
  output                  HPS_DDR3_CAS_n;
  output                  HPS_DDR3_CKE;
  output                  HPS_DDR3_CK_n;
  output                  HPS_DDR3_CK_p;
  output                  HPS_DDR3_CS_n;
  output [3:0]                    HPS_DDR3_DM;
  inout [31:0]                    HPS_DDR3_DQ;
  inout [3:0]                         HPS_DDR3_DQS_n;
  inout [3:0]                         HPS_DDR3_DQS_p;
```

```
output              HPS_DDR3_ODT;
output              HPS_DDR3_RAS_n;
output              HPS_DDR3_RESET_n;
input               HPS_DDR3_RZQ;
output              HPS_DDR3_WE_n;
input               HPS_ENET_GTX_CLK;
input               HPS_ENET_INT_n;
output              HPS_ENET_MDC;
inout               HPS_ENET_MDIO;
output              HPS_ENET_RESET_n;
input               HPS_ENET_RX_CLK;
input [3:0]                 HPS_ENET_RX_DATA;
input               HPS_ENET_RX_DV;
output [3:0]            HPS_ENET_TX_DATA;
output              HPS_ENET_TX_EN;
inout [3:0]                 HPS_FLASH_DATA;
output              HPS_FLASH_DCLK;
output              HPS_FLASH_NCSO;
input               HPS_GSENSOR_INT;
inout               HPS_I2C_CLK;
inout               HPS_I2C_SDA;
inout [3:0]                 HPS_KEY;
output              HPS_LCM_D_C;
output              HPS_LCM_RST_N;
input               HPS_LCM_SPIM_CLK;
inout               HPS_LCM_SPIM_MISO;
output              HPS_LCM_SPIM_MOSI;
output              HPS_LCM_SPIM_SS;
output [3:0]            HPS_LED;
```

```
  inout                       HPS_LTC_GPIO;
  input                       HPS_RESET_n;
  output                       HPS_SD_CLK;
  inout                       HPS_SD_CMD;
  inout [3:0]                                 HPS_SD_DATA;
  output                       HPS_SPIM_CLK;
  input                       HPS_SPIM_MISO;
  output                       HPS_SPIM_MOSI;
  output                       HPS_SPIM_SS;
  input [3:0]                                 HPS_SW;
  input                       HPS_UART_RX;
  output                       HPS_UART_TX;
  input                       HPS_USB_CLKOUT;
  inout [7:0]                                 HPS_USB_DATA;
  input                       HPS_USB_DIR;
  input                       HPS_USB_NXT;
  output                       HPS_USB_RESET_PHY;
  output                       HPS_USB_STP;
  input                       HPS_WARM_RST_n;
`endif /*ENABLE_HPS*/

  ///////// HSMC /////////
  input [2:1]                                 HSMC_CLKIN_n;
  input [2:1]                                 HSMC_CLKIN_p;
  output [2:1]                          HSMC_CLKOUT_n;
  output [2:1]                          HSMC_CLKOUT_p;
  input                       HSMC_CLK_IN0;
  output                       HSMC_CLK_OUT0;
  inout [3:0]                                 HSMC_D;
```

```
`ifdef ENABLE_HSMC_XCVR
  input [7:0]                       HSMC_GXB_RX_p;
  output [7:0]                      HSMC_GXB_TX_p;
  input                     HSMC_REF_CLK_p;
`endif
  inout [16:0]                      HSMC_RX_n;
  inout [16:0]                      HSMC_RX_p;
  output                     HSMC_SCL;
  inout                     HSMC_SDA;
  inout [16:0]                      HSMC_TX_n;
  inout [16:0]                      HSMC_TX_p;

  ///////// IRDA /////////
  input                     IRDA_RXD;

  ///////// KEY /////////
  input [3:0]                       KEY;

  ///////// LED /////////
  output [3:0]                      LED;

  ///////// OSC /////////
  input                     OSC_50_B3B;
  input                     OSC_50_B4A;
  input                     OSC_50_B5B;
  input                     OSC_50_B8A;

  ///////// PCIE /////////
  input                     PCIE_PERST_n;
```

```
input                       PCIE_WAKE_n;

//////// RESET ////////
input                       RESET_n;

//////// SI5338 ////////
inout                       SI5338_SCL;
inout                       SI5338_SDA;

//////// SW ////////
input [3:0]                             SW;

//////// TEMP ////////
output                       TEMP_CS_n;
output                       TEMP_DIN;
input                       TEMP_DOUT;
output                       TEMP_SCLK;

//////// USB ////////
input                       USB_B2_CLK;
inout [7:0]                             USB_B2_DATA;
output                       USB_EMPTY;
output                       USB_FULL;
input                       USB_OE_n;
input                       USB_RD_n;
input                       USB_RESET_n;
inout                       USB_SCL;
inout                       USB_SDA;
input                       USB_WR_n;
```

```
//////// VGA ////////
output [7:0]                    VGA_B;
output                          VGA_BLANK_n;
output                          VGA_CLK;
output [7:0]                    VGA_G;
output                          VGA_HS;
output [7:0]                    VGA_R;
output                          VGA_SYNC_n;
output                          VGA_VS;

////////hps pin///////
output wire [14:0]                      memory_mem_a;
output wire [2:0]                       memory_mem_ba;
output wire                             memory_mem_ck;
output wire                             memory_mem_ck_n;
output wire                             memory_mem_cke;
output wire                             memory_mem_cs_n;
output wire                             memory_mem_ras_n;
output wire                             memory_mem_cas_n;
output wire                             memory_mem_we_n;
output wire                             memory_mem_reset_n;
inout  wire [31:0]                      memory_mem_dq;
inout  wire [3:0]                       memory_mem_dqs;
inout  wire [3:0]                       memory_mem_dqs_n;
output wire                             memory_mem_odt;
output wire [3:0]                       memory_mem_dm;
input  wire                             memory_oct_rzqin;
output wire                             hps_io_hps_io_emac1_inst_TX_CLK;
```

```
output wire  hps_io_hps_io_emac1_inst_TXD0;
output wire  hps_io_hps_io_emac1_inst_TXD1;
output wire  hps_io_hps_io_emac1_inst_TXD2;
output wire  hps_io_hps_io_emac1_inst_TXD3;
input  wire  hps_io_hps_io_emac1_inst_RXD0;
inout  wire  hps_io_hps_io_emac1_inst_MDIO;
output wire  hps_io_hps_io_emac1_inst_MDC;
input  wire  hps_io_hps_io_emac1_inst_RX_CTL;
output wire  hps_io_hps_io_emac1_inst_TX_CTL;
input  wire  hps_io_hps_io_emac1_inst_RX_CLK;
input  wire  hps_io_hps_io_emac1_inst_RXD1;
input  wire  hps_io_hps_io_emac1_inst_RXD2;
input  wire  hps_io_hps_io_emac1_inst_RXD3;
inout  wire  hps_io_hps_io_qspi_inst_IO0;
inout  wire  hps_io_hps_io_qspi_inst_IO1;
inout  wire  hps_io_hps_io_qspi_inst_IO2;
inout  wire  hps_io_hps_io_qspi_inst_IO3;
output wire  hps_io_hps_io_qspi_inst_SS0;
output wire  hps_io_hps_io_qspi_inst_CLK;
inout  wire  hps_io_hps_io_sdio_inst_CMD;
inout  wire  hps_io_hps_io_sdio_inst_D0;
inout  wire  hps_io_hps_io_sdio_inst_D1;
output wire  hps_io_hps_io_sdio_inst_CLK;
inout  wire  hps_io_hps_io_sdio_inst_D2;
inout  wire  hps_io_hps_io_sdio_inst_D3;
inout  wire  hps_io_hps_io_usb1_inst_D0;
inout  wire  hps_io_hps_io_usb1_inst_D1;
inout  wire  hps_io_hps_io_usb1_inst_D2;
inout  wire  hps_io_hps_io_usb1_inst_D3;
```

```
inout  wire                                hps_io_hps_io_usb1_inst_D4;
inout  wire                                hps_io_hps_io_usb1_inst_D5;
inout  wire                                hps_io_hps_io_usb1_inst_D6;
inout  wire                                hps_io_hps_io_usb1_inst_D7;
input  wire                                hps_io_hps_io_usb1_inst_CLK;
output wire                                hps_io_hps_io_usb1_inst_STP;
input  wire                                hps_io_hps_io_usb1_inst_DIR;
input  wire                                hps_io_hps_io_usb1_inst_NXT;
output wire                                hps_io_hps_io_spim0_inst_CLK;
output wire                                hps_io_hps_io_spim0_inst_MOSI;
input  wire                                hps_io_hps_io_spim0_inst_MISO;
output wire                                hps_io_hps_io_spim0_inst_SS0;
output wire                                hps_io_hps_io_spim1_inst_CLK;
output wire                                hps_io_hps_io_spim1_inst_MOSI;
input  wire                                hps_io_hps_io_spim1_inst_MISO;
output wire                                hps_io_hps_io_spim1_inst_SS0;
input  wire                                hps_io_hps_io_uart0_inst_RX;
output wire                                hps_io_hps_io_uart0_inst_TX;
inout  wire                                hps_io_hps_io_i2c1_inst_SDA;
inout  wire                                hps_io_hps_io_i2c1_inst_SCL;
inout  wire                                hps_io_hps_io_gpio_inst_GPIO00;
//=======================================================
//  REG/WIRE declarations
//=======================================================

//      For Audio CODEC
wire                                AUD_CTRL_CLK;       //        For Audio Controller

reg [31:0]                                Cont;
```

```
wire                            VGA_CTRL_CLK;
wire [9:0]                      mVGA_R;
wire [9:0]                      mVGA_G;
wire [9:0]                      mVGA_B;
wire [19:0]                     mVGA_ADDR;
wire                            DLY_RST;

//      For VGA Controller
wire                            mVGA_CLK;
wire [9:0]                      mRed;
wire [9:0]                      mGreen;
wire [9:0]                      mBlue;
wire                            VGA_Read;  //        VGA data request

wire [9:0]                      recon_VGA_R;
wire [9:0]                      recon_VGA_G;
wire [9:0]                      recon_VGA_B;

//      For Down Sample
wire [3:0]                      Remain;
wire [9:0]                      Quotient;

wire                            AUD_MUTE;

// Drive the LEDs with the switches
//assign LED = SW;

// Make the FPGA reset cause an HPS reset
reg [19:0]                      hps_reset_counter = 20'h0;
```

```
  reg                                                    hps_fpga_reset_n = 0;

//AUDIO
wire reset = !KEY[0];
wire main_clk;
wire audio_clk;

wire [1:0] sample_end;
wire [1:0] sample_req;
wire [15:0] audio_output;
wire [15:0] audio_input;
wire [15:0] sound_new;
wire [14:0] addr;

city b1
            ( .address(addr),
              .clock(OSC_50_B8A),
              .q(sound_new));
clock_pll pll (
   .refclk (OSC_50_B8A),
   .rst (reset),
   .outclk_0 (audio_clk),
   .outclk_1 (main_clk)
);

i2c_av_config av_config (
   .clk (main_clk),
   .reset (reset),
   .i2c_sclk (AUD_I2C_SCLK),
```

```
    .i2c_sdat (AUD_I2C_SDAT),
    .status (LED)
);

assign AUD_XCK = audio_clk;
assign AUD_MUTE = (SW != 4'b0);

audio_codec ac (
    .clk (audio_clk),
    .reset (reset),
    .sample_end (sample_end),
    .sample_req (sample_req),
    .audio_output (audio_output),
    .audio_input (audio_input),
    .channel_sel (2'b10),

    .AUD_ADCLRCK (AUD_ADCLRCK),
    .AUD_ADCDAT (AUD_ADCDAT),
    .AUD_DACLRCK (AUD_DACLRCK),
    .AUD_DACDAT (AUD_DACDAT),
    .AUD_BCLK (AUD_BCLK)
);

audio_effects ae (
    .clk (audio_clk),
    .sample_end (sample_end[1]),
    .sample_req (sample_req[1]),
    .audio_output (audio_output),
    .audio_input  (audio_input),
```

```
 .control (SW),
          .sound_new(sound_new),
          .addr(addr),
         );
always @(posedge OSC_50_B4A) begin
  if (hps_reset_counter == 20'h ffffff) hps_fpga_reset_n <= 1;
  hps_reset_counter <= hps_reset_counter + 1;
end

lab3 u0 (
        .clk_clk                       (OSC_50_B4A),                //            clk.clk
        .reset_reset_n                 (hps_fpga_reset_n),          //          reset.reset_n
        .memory_mem_a                  (memory_mem_a),              //         memory.mem_a
        .memory_mem_ba                 (memory_mem_ba),             //           .mem_ba
        .memory_mem_ck                 (memory_mem_ck),             //           .mem_ck
        .memory_mem_ck_n               (memory_mem_ck_n),           //           .mem_ck_n
        .memory_mem_cke                (memory_mem_cke),            //           .mem_cke
        .memory_mem_cs_n               (memory_mem_cs_n),           //           .mem_cs_n
        .memory_mem_ras_n              (memory_mem_ras_n),          //           .mem_ras_n
        .memory_mem_cas_n              (memory_mem_cas_n),          //           .mem_cas_n
        .memory_mem_we_n               (memory_mem_we_n),           //           .mem_we_n
        .memory_mem_reset_n            (memory_mem_reset_n),        //           .mem_reset_n
        .memory_mem_dq                 (memory_mem_dq),             //           .mem_dq
        .memory_mem_dqs                (memory_mem_dqs),            //           .mem_dqs
        .memory_mem_dqs_n              (memory_mem_dqs_n),          //           .mem_dqs_n
        .memory_mem_odt                (memory_mem_odt),            //           .mem_odt
        .memory_mem_dm                 (memory_mem_dm),             //           .mem_dm
        .memory_oct_rzqin              (memory_oct_rzqin),          //           .oct_rzqin
```

```
        .hps_io_hps_io_emac1_inst_TX_CLK                (hps_io_hps_io_emac1_inst_TX_CLK), //
                                .hps_0_hps_io.hps_io_emac1_inst_TX_CLK
        .hps_io_hps_io_emac1_inst_TXD0                  (hps_io_hps_io_emac1_inst_TXD0),   //
.hps_io_emac1_inst_TXD0
        .hps_io_hps_io_emac1_inst_TXD1                  (hps_io_hps_io_emac1_inst_TXD1),   //
.hps_io_emac1_inst_TXD1
        .hps_io_hps_io_emac1_inst_TXD2                  (hps_io_hps_io_emac1_inst_TXD2),   //
.hps_io_emac1_inst_TXD2
        .hps_io_hps_io_emac1_inst_TXD3                  (hps_io_hps_io_emac1_inst_TXD3),   //
.hps_io_emac1_inst_TXD3
        .hps_io_hps_io_emac1_inst_RXD0                  (hps_io_hps_io_emac1_inst_RXD0),   //
.hps_io_emac1_inst_RXD0
        .hps_io_hps_io_emac1_inst_MDIO                  (hps_io_hps_io_emac1_inst_MDIO),   //
.hps_io_emac1_inst_MDIO
        .hps_io_hps_io_emac1_inst_MDC                   (hps_io_hps_io_emac1_inst_MDC),    //
.hps_io_emac1_inst_MDC
        .hps_io_hps_io_emac1_inst_RX_CTL                (hps_io_hps_io_emac1_inst_RX_CTL), //
.hps_io_emac1_inst_RX_CTL
        .hps_io_hps_io_emac1_inst_TX_CTL                (hps_io_hps_io_emac1_inst_TX_CTL), //
.hps_io_emac1_inst_TX_CTL
        .hps_io_hps_io_emac1_inst_RX_CLK                (hps_io_hps_io_emac1_inst_RX_CLK), //
.hps_io_emac1_inst_RX_CLK
        .hps_io_hps_io_emac1_inst_RXD1                  (hps_io_hps_io_emac1_inst_RXD1),   //
.hps_io_emac1_inst_RXD1
        .hps_io_hps_io_emac1_inst_RXD2                  (hps_io_hps_io_emac1_inst_RXD2),   //
.hps_io_emac1_inst_RXD2
        .hps_io_hps_io_emac1_inst_RXD3                  (hps_io_hps_io_emac1_inst_RXD3),   //
.hps_io_emac1_inst_RXD3
        .hps_io_hps_io_qspi_inst_IO0                (hps_io_hps_io_qspi_inst_IO0),     //
.hps_io_qspi_inst_IO0
        .hps_io_hps_io_qspi_inst_IO1                (hps_io_hps_io_qspi_inst_IO1),     //
.hps_io_qspi_inst_IO1
        .hps_io_hps_io_qspi_inst_IO2                (hps_io_hps_io_qspi_inst_IO2),     //
.hps_io_qspi_inst_IO2
```

```
        .hps_io_hps_io_qspi_inst_IO3               (hps_io_hps_io_qspi_inst_IO3),    //
.hps_io_qspi_inst_IO3

        .hps_io_hps_io_qspi_inst_SS0               (hps_io_hps_io_qspi_inst_SS0),    //
.hps_io_qspi_inst_SS0

        .hps_io_hps_io_qspi_inst_CLK               (hps_io_hps_io_qspi_inst_CLK),    //
.hps_io_qspi_inst_CLK

        .hps_io_hps_io_sdio_inst_CMD                    (hps_io_hps_io_sdio_inst_CMD),    //
.hps_io_sdio_inst_CMD

        .hps_io_hps_io_sdio_inst_D0                (hps_io_hps_io_sdio_inst_D0),     //
.hps_io_sdio_inst_D0

        .hps_io_hps_io_sdio_inst_D1                (hps_io_hps_io_sdio_inst_D1),     //
.hps_io_sdio_inst_D1

        .hps_io_hps_io_sdio_inst_CLK               (hps_io_hps_io_sdio_inst_CLK),    //
.hps_io_sdio_inst_CLK

        .hps_io_hps_io_sdio_inst_D2                (hps_io_hps_io_sdio_inst_D2),     //
.hps_io_sdio_inst_D2

        .hps_io_hps_io_sdio_inst_D3                (hps_io_hps_io_sdio_inst_D3),     //
.hps_io_sdio_inst_D3

        .hps_io_hps_io_usb1_inst_D0                (hps_io_hps_io_usb1_inst_D0),     //
.hps_io_usb1_inst_D0

        .hps_io_hps_io_usb1_inst_D1                (hps_io_hps_io_usb1_inst_D1),     //
.hps_io_usb1_inst_D1

        .hps_io_hps_io_usb1_inst_D2                (hps_io_hps_io_usb1_inst_D2),     //
.hps_io_usb1_inst_D2

        .hps_io_hps_io_usb1_inst_D3                (hps_io_hps_io_usb1_inst_D3),     //
.hps_io_usb1_inst_D3

        .hps_io_hps_io_usb1_inst_D4                (hps_io_hps_io_usb1_inst_D4),     //
.hps_io_usb1_inst_D4

        .hps_io_hps_io_usb1_inst_D5                (hps_io_hps_io_usb1_inst_D5),     //
.hps_io_usb1_inst_D5

        .hps_io_hps_io_usb1_inst_D6                (hps_io_hps_io_usb1_inst_D6),     //
.hps_io_usb1_inst_D6

        .hps_io_hps_io_usb1_inst_D7                (hps_io_hps_io_usb1_inst_D7),     //
.hps_io_usb1_inst_D7
```

```
        .hps_io_hps_io_usb1_inst_CLK        (hps_io_hps_io_usb1_inst_CLK),    //
.hps_io_usb1_inst_CLK
        .hps_io_hps_io_usb1_inst_STP        (hps_io_hps_io_usb1_inst_STP),    //
.hps_io_usb1_inst_STP
        .hps_io_hps_io_usb1_inst_DIR        (hps_io_hps_io_usb1_inst_DIR),    //
.hps_io_usb1_inst_DIR
        .hps_io_hps_io_usb1_inst_NXT        (hps_io_hps_io_usb1_inst_NXT),    //
.hps_io_usb1_inst_NXT
        .hps_io_hps_io_spim0_inst_CLK           (hps_io_hps_io_spim0_inst_CLK),   //
.hps_io_spim0_inst_CLK
        .hps_io_hps_io_spim0_inst_MOSI          (hps_io_hps_io_spim0_inst_MOSI),  //
.hps_io_spim0_inst_MOSI
        .hps_io_hps_io_spim0_inst_MISO          (hps_io_hps_io_spim0_inst_MISO),  //
.hps_io_spim0_inst_MISO
        .hps_io_hps_io_spim0_inst_SS0           (hps_io_hps_io_spim0_inst_SS0),   //
.hps_io_spim0_inst_SS0
        .hps_io_hps_io_spim1_inst_CLK           (hps_io_hps_io_spim1_inst_CLK),   //
.hps_io_spim1_inst_CLK
        .hps_io_hps_io_spim1_inst_MOSI          (hps_io_hps_io_spim1_inst_MOSI),  //
.hps_io_spim1_inst_MOSI
        .hps_io_hps_io_spim1_inst_MISO          (hps_io_hps_io_spim1_inst_MISO),  //
.hps_io_spim1_inst_MISO
        .hps_io_hps_io_spim1_inst_SS0           (hps_io_hps_io_spim1_inst_SS0),   //
.hps_io_spim1_inst_SS0
        .hps_io_hps_io_uart0_inst_RX        (hps_io_hps_io_uart0_inst_RX),    //
.hps_io_uart0_inst_RX
        .hps_io_hps_io_uart0_inst_TX        (hps_io_hps_io_uart0_inst_TX),    //
.hps_io_uart0_inst_TX
        .hps_io_hps_io_i2c1_inst_SDA        (hps_io_hps_io_i2c1_inst_SDA),    //
.hps_io_i2c1_inst_SDA
        .hps_io_hps_io_i2c1_inst_SCL        (hps_io_hps_io_i2c1_inst_SCL),    //
.hps_io_i2c1_inst_SCL
                                .vga_R (VGA_R),
                                .vga_G (VGA_G),
```

```
                                .vga_B (VGA_B),
                                .vga_CLK (VGA_CLK),
                                .vga_HS (VGA_HS),
                                .vga_VS (VGA_VS),
                                .vga_BLANK_n (VGA_BLANK_n),
                                .vga_SYNC_n (VGA_SYNC_n),

        );

endmodule
```

VGA_LED.sv

```
/*
 * Avalon memory-mapped peripheral for the VGA LED Emulator
 *
 * Stephen A. Edwards
 * Columbia University
 */

module VGA_LED(input logic       clk,
               input logic     reset,
               input logic [7:0]  writedata,
               input logic     write,
               input                   chipselect,
               input logic [2:0]  address,

               output logic [7:0] VGA_R, VGA_G, VGA_B,
               output logic            VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_n,
               output logic            VGA_SYNC_n);
```

```
logic [7:0]                     hex0, hex1, hex2, hex3,
                                hex4, hex5, hex6, hex7;

VGA_LED_Emulator led_emulator(.clk50(clk), .*);

always_ff @(posedge clk)
  if (reset) begin
        /*hex0 <= 8'b01100110; // 4
        hex1 <= 8'b01111111; // 8
        hex2 <= 8'b01100110; // 4
        hex3 <= 8'b10111111; // 0
        hex4 <= 8'b00111000; // L
        hex5 <= 8'b01110111; // A
        hex6 <= 8'b01111100; // b
        hex7 <= 8'b01001111; // 3*/
        hex0 <= 8'b00100001;
        hex1 <= 8'b00000011;
hex3 <= 8'b00000000;
hex5 <= 8'b00000001;
hex4 <= 8'b00111100;
        //hex2 <= 8'b01100110;
        //hex3 <= 8'b10111111;
        //hex4 <= 8'b00111000;
        //hex5 <= 8'b01110111;
        //hex6 <= 8'b01111100;
        //hex7 <= 8'b01001111;
        //x <= 16'd300;
```

```
        //y <= 16'd300;
         end else if (chipselect && write) begin
    case (address)
        4'h0 : hex0 <= writedata;
        4'h1 : hex1 <= writedata;
        4'h2 : hex2 <= writedata;
        4'h3 : hex3 <= writedata;
        4'h4 : hex4 <= writedata;
        4'h5 : hex5 <= writedata;
        4'h6 : hex6 <= writedata;
        4'h7 : hex7 <= writedata;
    endcase

                end

endmodule
```

VGA_LED_Emulator.sv

```
/*
 * Seven-segment LED emulator
 *
 * Stephen A. Edwards, Columbia University
 */

module VGA_LED_Emulator(
 input logic           clk50, reset,
 input logic [7:0]  hex0, hex1, hex2, hex3, hex4, hex5, hex6, hex7,
 //input logic [15:0]  x, //centre of circle coordinates
 //input logic [15:0]  y,
```

```
output logic [7:0] VGA_R, VGA_G, VGA_B,
output logic       VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_n, VGA_SYNC_n);

/*
 * 640 X 480 VGA timing for a 50 MHz clock: one pixel every other cycle
 *
 * HCOUNT 1599 0           1279      1599 0
 *         _________________          ________
 * ____________|    Video     |_____________|  Video
 *
 *
 * |SYNC| BP |<-- HACTIVE -->|FP|SYNC| BP |<-- HACTIVE
 *      ____________________________     _______________
 * |____|      VGA_HS        |____|
 */
  // Parameters for hcount
  parameter HACTIVE      = 11'd 1280,
            HFRONT_PORCH = 11'd 32,
            HSYNC        = 11'd 192,
            HBACK_PORCH  = 11'd 96,
            HTOTAL       = HACTIVE + HFRONT_PORCH + HSYNC + HBACK_PORCH; // 1600

  // Parameters for vcount
  parameter VACTIVE      = 10'd 480,
            VFRONT_PORCH = 10'd 10,
            VSYNC        = 10'd 2,
            VBACK_PORCH  = 10'd 33,
            VTOTAL       = VACTIVE + VFRONT_PORCH + VSYNC + VBACK_PORCH; // 525
```

```
        // Parameter for vcircle

logic [10:0]                    hcount; // Horizontal counter
                                // Hcount[10:1] indicates pixel column (0-639)
logic                           endOfLine;

        wire pclk;
        assign pclk = clk50*50;

        // Moles & Hammer
        logic [9:0] mole1_x,mole2_x,mole3_x,hammer_x;
        logic [9:0] mole1_y,mole2_y,mole3_y,hammer_y;
        logic [9:0] temp_mole1_x,temp_mole2_x,temp_mole3_x;
        logic [9:0] temp_mole1_y,temp_mole2_y,temp_mole3_y;

        logic[75:0][63:0] shape2mem,shape3mem,shape4mem,shape5mem,shape6mem,shape7mem;

        logic [6:0] mole1pop=0,mole2pop=0,mole3pop=0;



        //Grass
        logic [7:0][7:0] grassmem;
        logic [29:0][63:0] grass1mem;
        logic [39:0][63:0] grass2mem;

        //Stones
        logic [51:0][119:0] stonemem;
```

```
        //Mud
        logic [18:0][119:0] mudmem;



        //Score
        logic [74:0][159:0] scoreboardmem;
        logic [78:0][159:0] scoreboard1mem;

        logic [3:0] score0,score1,score2;
        //LEVEL
        logic[15:0][95:0] levelmem;
        logic [3:0]  level0,level1;

        //LIVES
         logic [15:0][63:0] livesmem;

        //SCORE
        logic [15:0][117:0] scoremem;

        //TIME
        logic [15:0][101:0] timemem;
        logic [3:0]  time0,time1;
        //WHAC-A-MOLE
        logic [40:0][63:0] momem;
logic [40:0][63:0] momem2;
logic [40:0][63:0] momem3;
logic [40:0][63:0] mememc;
logic [40:0][63:0] mmem;
logic [40:0][63:0] omem;
```

```
  logic [40:0][63:0] lmem;
  logic [40:0][63:0] emem;
  logic [15:0][15:0] dash;

        //HAMMER
        logic[96:0][96:0] hammermem, hammer2mem;
        logic[31:0][96:0] hammer3mem;

        ///NUMBERS
        logic [31:0][31:0] zero;
  logic [31:0][31:0] one;
  logic [31:0][31:0] two;
  logic [31:0][31:0] three;
        logic [31:0][31:0] four;
        logic [31:0][31:0] five;
  logic [31:0][31:0] six;
  logic [31:0][31:0] seven;
  logic [31:0][31:0] eight;
        logic [31:0][31:0] nine;

        /*//GEORGE'S HEX DIGIT

        logic [9:0] d_l_1_x,d_l_1_y;
        assign d_l_1_y = 30;
        assign d_l_1_x = 15;
        logic d_l_1_a,d_l_1_b,d_l_1_c,d_l_1_d,d_l_1_e,d_l_1_f,d_l_1_g;

        assign d_l_1_a = ((hcount[10:1]>=d_l_1_x) & (hcount[10:1]<=d_l_1_x+31) & (vcount>=d_l_1_y)
& (vcount<=d_l_1_y+9)) ;
```

```
  assign d_l_1_d = ((hcount[10:1]>=d_l_1_x) & (hcount[10:1]<=d_l_1_x+31) & (vcount>=d_l_1_y+50) &
(vcount<=d_l_1_y+59)) ;

        assign d_l_1_g = ((hcount[10:1]>=d_l_1_x) & (hcount[10:1]<=d_l_1_x+31) &
(vcount>=d_l_1_y+25) & (vcount<=d_l_1_y+34)) ;

        assign d_l_1_b = ((hcount[10:1]>=d_l_1_x+22) & (hcount[10:1]<=d_l_1_x+31) &
(vcount>=d_l_1_y) & (vcount<=d_l_1_y+39)) ;

        assign d_l_1_c = ((hcount[10:1]>=d_l_1_x+22) & (hcount[10:1]<=d_l_1_x+31) &
(vcount>=d_l_1_y+25) & (vcount<=d_l_1_y+59)) ;

  assign d_l_1_f = ((hcount[10:1]>=d_l_1_x) & (hcount[10:1]<=d_l_1_x+9) & (vcount>=d_l_1_y) &
(vcount<=d_l_1_y+39)) ;

  assign d_l_1_e = ((hcount[10:1]>=d_l_1_x) & (hcount[10:1]<=d_l_1_x+9) & (vcount>=d_l_1_y+25) &
(vcount<=d_l_1_y+59)) ;*/


        //ROCKS

        //logic[19:0][19:0] rockmem,rockmem2;



  always_ff @(posedge clk50 or posedge reset)

   if (reset)        hcount <= 0;

   else if (endOfLine) hcount <= 0;

   else          hcount <= hcount + 11'd 1;


  assign endOfLine = hcount == HTOTAL - 1;


  // Vertical counter

  logic [9:0]                   vcount;

  logic                         endOfField;



  always_ff @(posedge clk50 or posedge reset)

   if (reset)        vcount <= 0;
```

```
    else if (endOfLine)
      if (endOfField)   vcount <= 0;
      else              vcount <= vcount + 10'd 1;

  assign endOfField = vcount == VTOTAL - 1;


always_ff @(posedge endOfField)begin
            if (hex0[3:0]!=0)
              if (mole1pop!=59) mole1pop<=mole1pop+1;
              else mole1pop<=59;
   else if (mole1pop!=0) mole1pop<=mole1pop-1;
                else mole1pop<=0;

                              if (hex0[3:0]!=0) begin
                                        temp_mole1_x<=mole1_x;
          temp_mole1_y<=mole1_y;
                                        end

  if (hex0[7:4]!=0)
            if (mole2pop!=59) mole2pop<=mole2pop+1;
            else mole2pop<=59;
  else if (mole2pop!=0) mole2pop<=mole2pop-1;
                else mole2pop<=0;

                if (hex0[7:4]!=0) begin
                                        temp_mole2_x<=mole2_x;
                                        temp_mole2_y<=mole2_y;
                                        end
```

```
          if (hex1[3:0]!=0)
           if (mole3pop!=59) mole3pop<=mole3pop+1;
           else mole3pop<=59;
 else if (mole3pop!=0) mole3pop<=mole3pop-1;
              else mole3pop<=0;

    if (hex1[3:0]!=0) begin
                                    temp_mole3_x<=mole3_x;
       temp_mole3_y<=mole3_y;
    end

end

// Horizontal sync: from 0x520 to 0x5DF (0x57F)
// 101 0010 0000 to 101 1101 1111
assign VGA_HS = !( (hcount[10:8] == 3'b101) & !(hcount[7:5] == 3'b111));
assign VGA_VS = !( vcount[9:1] == (VACTIVE + VFRONT_PORCH) / 2);

assign VGA_SYNC_n = 1; // For adding sync to video signals; not used for VGA

// Horizontal active: 0 to 1279     Vertical active: 0 to 479
// 101 0000 0000  1280               01 1110 0000  480
// 110 0011 1111  1599               10 0000 1100  524
assign VGA_BLANK_n = !( hcount[10] & (hcount[9] | hcount[8]) ) &
                            !( vcount[9] | (vcount[8:5] == 4'b1111) );

/* VGA_CLK is 25 MHz
```

```
 *          __    __    __
 * clk50   __|  |__|  |__|
 *
 *          _____      __
 * hcount[0]__|     |_____|
 */
assign VGA_CLK = hcount[0]; // 25 MHz clock: pixel latched on rising edge



        // this is bagri trying some cool shit


        always_comb begin


                            shape2mem[0] =
64'b0000000000000000000000000001111111000000000000000000000000000000;
                            shape2mem[1] =
64'b0000000000000000000001111111111111111111110000000000000000000000;
                            shape2mem[2] =
64'b0000000000000001111111111111111111111111111111110000000000000000;
                            shape2mem[3] =
64'b0000000000111111111111111111111111111111111111111111110000000000;
                            shape2mem[4] =
64'b0000001111111111111111111111111111111111111111111111111111000000;
                            shape2mem[5] =
64'b0001111111111111111111111111111111111111111111111111111111111000;
                            shape2mem[6] =
64'b0111111111111111111111111111111111111111111111111111111111111110;
                            shape2mem[7] =
64'b1111111111111111111111111111111111111111111111111111111111111111;
                        shape2mem[8] =
64'b1111111111111111111111111111111111111111111111111111111111111111;
```

shape2mem[9] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[10] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[11] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[12] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[13] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[14] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[15] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[16] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[17] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[18] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[19] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[20] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[21] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[22] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[23] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[24] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[25] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

```
                shape2mem[26] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[27] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[28] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[29] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[30] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[31] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[32] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[33] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[34] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[35] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[36] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[37] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[38] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

              shape2mem[39] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[40] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[41] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[42] =
64'b1111111111111111111111111111111111111111111111111111111111111111;
```

shape2mem[43] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[44] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[45] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[46] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[47] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[48] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[49] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[50] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[51] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[52] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[53] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[54] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[55] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[56] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[57] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[58] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape2mem[59] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

```
                shape2mem[60] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[61] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[62] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[63] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[64] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[65] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[66] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[67] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[68] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                shape2mem[69] =
64'b0001111111111111111111111111111111111111111111111111111111111000;

                shape2mem[70] =
64'b0000001111111111111111111111111111111111111111111111111111000000;

                shape2mem[71] =
64'b0000000001111111111111111111111111111111111111111111111000000000;

                shape2mem[72] =
64'b0000000000001111111111111111111111111111111111111110000000000000;

                shape2mem[73] =
64'b0000000000000001111111111111111111111111111111110000000000000000;

                shape2mem[74] =
64'b0000000000000000001111111111111111111111111100000000000000000000;

                shape2mem[75] =
64'b0000000000000000000001111111111111111111110000000000000000000000;


                //GLASSES
```

```
shape3mem[0] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[1] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[2] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[3] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[4] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[5] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[6] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[7] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
shape3mem[8] = 64'b1111111100000000000111111111111111111111110000000000001111111111;
shape3mem[9] = 64'b1111100000000000000000111111111111111111000000000000000001111111;
shape3mem[10] = 64'b1110000000000000000000011111111111111000000000000000000000011111;
shape3mem[11] = 64'b1110000000011111110000000111111111111000000001111111110000000111;
shape3mem[12] = 64'b1100000001111111111000000111111111100000001111111111100000011;
shape3mem[13] = 64'b1000000011111111111100000011111111000000111111111111100000001;
shape3mem[14] = 64'b1000000111111111111100000000000000000011111111111111110000000;
shape3mem[15] = 64'b0000001111111111111110000000000000001111111111111111110000000;
shape3mem[16] = 64'b0000000111111111111111000000000000111111111111111111100000000;
```

```
                    shape3mem[17] =
64'b0000000111111111111111110000000000011111111111111111111100000000;

                shape3mem[18] =
64'b1000000011111111111111110000000000011111111111111111111000000001;

                    shape3mem[19] =
64'b1000000011111111111111110000000000011111111111111111111000000001;

                    shape3mem[20] =
64'b1100000001111111111111110000000000011111111111111111111100000000011;

                    shape3mem[21] =
64'b1110000000111111111111000000000000000111111111111111000000000000111;

                    shape3mem[22] =
64'b1111000000000000000000000000011000000000000000000000000000000001111;

                    shape3mem[23] =
64'b1111100000000000000000000011111111000000000000000000000001111111111;

                    shape3mem[24] =
64'b1111111111110000000001111111111111111100000000001111111111111111;

                    shape3mem[25] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[26] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[27] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[28] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[29] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[30] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[31] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[32] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[33] =
64'b1111111111111111111111111111111111111111111111111111111111111111;
```

```
                    shape3mem[34] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[35] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[36] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[37] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[38] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                  shape3mem[39] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[40] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[41] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[42] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[43] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[44] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[45] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[46] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[47] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[48] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[49] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

                    shape3mem[50] =
64'b1111111111111111111111111111111111111111111111111111111111111111;
```

shape3mem[51] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[52] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[53] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[54] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[55] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[56] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[57] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[58] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[59] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[60] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[61] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[62] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[63] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[64] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[65] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[66] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[67] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[68] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[69] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[70] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[71] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[72] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[73] =
64'b1111111111111111111111111111111111111111111111101111111111111111;

shape3mem[74] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

shape3mem[75] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

//WHITE OF THE EYES

shape4mem[0] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[1] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[2] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[3] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[4] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[5] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[6] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[7] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[8] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[9] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[10] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[11] =
64'b0000000000011111110000000000000000000000000011111111100000000000;

shape4mem[12] =
64'b0000000001111111111100000000000000000000001111111111110000000000;

shape4mem[13] =
64'b0000000011111111111111000000000000000000011111111111111100000000;

shape4mem[14] =
64'b0000000111111111111111000000000000000000111111111111111110000000;

shape4mem[15] =
64'b0000001111111111111111100000000000000001111111111111111111000000;

shape4mem[16] =
64'b0000001111111111111111100000000000000111111111111111111111000000;

shape4mem[17] =
64'b0000000111111111000000100000000000000001111111111111111100000000;

shape4mem[18] =
64'b0000000011111110000000000000000000000000111111111111000000000000;

shape4mem[19] =
64'b0000000011111000000000000000000000000000111111111111000000000000;

shape4mem[20] =
64'b0000000001111100000000000000000000000000111111111100000000000000;

shape4mem[21] =
64'b0000000000111111000000000000000000000000111111111000000000000000;

shape4mem[22] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[23] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[24] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[25] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[26] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[27] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[28] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[29] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[30] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[31] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[32] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[33] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[34] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[35] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[36] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[37] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[38] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[39] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[40] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[41] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[42] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[43] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[44] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[45] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[46] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[47] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[48] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[49] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[50] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[51] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[52] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[53] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[54] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[55] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[56] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[57] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[58] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[59] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[60] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[61] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[62] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[63] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[64] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[65] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[66] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[67] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[68] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[69] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[70] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[71] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[72] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[73] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[74] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape4mem[75] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

//NOSE

shape5mem[0] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[1] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[2] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[3] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[4] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[5] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[6] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[7] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[8] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[9] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[10] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[11] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[12] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[13] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[14] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[15] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[16] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[17] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[18] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[19] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[20] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[21] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[22] =
64'b0000000000000000000000000001111000000000000000000000000000000000;

shape5mem[23] =
64'b0000000000000000000000001111111111100000000000000000000000000000;

shape5mem[24] =
64'b0000000000000000000001111111111111111100000000000000000000000000;

shape5mem[25] =
64'b0000000000000000000011111111111111111110000000000000000000000000;

shape5mem[26] =
64'b0000000000000000000011111111111111111110000000000000000000000000;

shape5mem[27] =
64'b0000000000000000000011111111111111100000000000000000000000000000;

shape5mem[28] =
64'b0000000000000000000001111111111000000000000000000000000000000000;

shape5mem[29] =
64'b0000000000000000000000111111110000000000000000000000000000000000;

shape5mem[30] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[31] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[32] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[33] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[34] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[35] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[36] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[37] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[38] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[39] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[40] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[41] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[42] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[43] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[44] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[45] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[46] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[47] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[48] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[49] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[50] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[51] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[52] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[53] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[54] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[55] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[56] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[57] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[58] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[59] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[60] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[61] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[62] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[63] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[64] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[65] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[66] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[67] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[68] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[69] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[70] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[71] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[72] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[73] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[74] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape5mem[75] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

//EYES

shape6mem[0] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[1] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[2] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[3] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[4] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[5] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[6] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[7] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[8] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[9] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[10] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[11] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[12] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[13] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[14] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[15] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[16] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[17] =
64'b0000000000000000111111100000000000111111100000000000000000000000;

shape6mem[18] =
64'b0000000000000001111111110000000000111111111000000000000000000000;

shape6mem[19] =
64'b0000000000000111111111110000000001111111111000000000000000000000;

shape6mem[20] =
64'b0000000000000011111111100000000000111111111000000000000000000000;

shape6mem[21] =
64'b0000000000000000111110000000000000001111100000000000000000000000;

shape6mem[22] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[23] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[24] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[25] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[26] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[27] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[28] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[29] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[30] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[31] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[32] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[33] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[34] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[35] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[36] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[37] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[38] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[39] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[40] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[41] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[42] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[43] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[44] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[45] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[46] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[47] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[48] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[49] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[50] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[51] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[52] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[53] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[54] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[55] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[56] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[57] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[58] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[59] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[60] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[61] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[62] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[63] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[64] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[65] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[66] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[67] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[68] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[69] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[70] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[71] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[72] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[73] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[74] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape6mem[75] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

//TEETH

shape7mem[0] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[1] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[2] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[3] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[4] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[5] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[6] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[7] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[8] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[9] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[10] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[11] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[12] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[13] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[14] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[15] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[16] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[17] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[18] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[19] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[20] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[21] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[22] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[23] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[24] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[25] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[26] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[27] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[28] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[29] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[30] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[31] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[32] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[33] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[34] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[35] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[36] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[37] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[38] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[39] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[40] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[41] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[42] =
64'b0000000000011111111111111111111001111111111111111111100000000000;

shape7mem[43] =
64'b0000000000001111111111111111111000011111111111111111100000000000;

shape7mem[44] =
64'b0000000000000011111111111111111000000111111111111111100000000000;

shape7mem[45] =
64'b0000000000000001111111111111110000000011111111111111100000000000;

shape7mem[46] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[47] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[48] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[49] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[50] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[51] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[52] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[53] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[54] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[55] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[56] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[57] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[58] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[59] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[60] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[61] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[62] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[63] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[64] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[65] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[66] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[67] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[68] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[69] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[70] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[71] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[72] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[73] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[74] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

shape7mem[75] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

//LEVEL

levelmem[0] =
96'b111111000000000011111111111110001111110000000000100011111111111110001111110000000000000000000000000;

levelmem[1] =
96'b111111000000000011111100000000001111110000000000100011111100000000000111111000000000000000000000000;

levelmem[2] =
96'b111111000000000011111100000000000111111000000001000011111100000000000111111000000000000000000000000;

levelmem[3] =
96'b111111000000000011111100000000000111111000000001000011111100000000000111111000000000000000000000000;

levelmem[4] =
96'b111111000000000011111100000000000111111000000100000111111000000000000111111000000000000000000000000;

levelmem[5] =
96'b111111000000000011111111111110000011111100000010000011111111111110001111110000000000000000000000000;

levelmem[6] =
96'b111111000000000011111100000000000011111100001000000111111000000000000111111000000000000000000000000;

levelmem[7] =
96'b111111000000000011111100000000000011111100001000000111111000000000000111111000000000000000000000000;

levelmem[8] =
96'b111111000000000011111100000000000011111100100000001111110000000000011111100000000000000000000000000;

levelmem[9] =
96'b111111000000000011111100000000000011111100100000001111110000000000011111100000000000000000000000000;

levelmem[10] =
96'b111111000000000011111100000000000011111110000000011111100000000000111111000000000000000000000000000;

levelmem[11] =
96'b111111000000000011111100000000000011111110000000011111100000000000111111000000000000000000000000000;

levelmem[12] =
96'b111111000000000011111100000000000011111000000000111111000000000001111110000000000000000000000000000;

levelmem[13] =
96'b111111000000000011111100000000000011111000000000111111000000000001111110000000000000000000000000000;

levelmem[14] =
96'b111111000000000011111100000000000011100000000000111111000000000001111110000000000000000000000000000;

levelmem[15] =
96'b111111111111100011111111111110000000001110000000001111111111110001111111111111000000000000000;

//SCORE

scoremem[0] =
118'b000000000001110000000000000000000011100000000000000000000001110000000000000000111111111110000000000000011111111111111;

scoremem[1] =
118'b000000001110001110000000000000011100011100000000000000011111110001110000000000011111111000011100000000000011111111111111;

scoremem[2] =
118'b000001111100000001110000000011100000000011100000000111111111100000001110000000011111111000000000011100011111111000000;

scoremem[3] =
118'b001111111100000000000000011111000000000000000000111111111111110000000001110000011111111000000000011100111111111000000;

scoremem[4] =
118'b111111111111000000000000111111000000000000000000111111111111110000000001110000011111111000000000011100111111111000000;

scoremem[5] =
118'b111111111111100000000001111111000000000000000000111111111111110000000000111000011111111000000000011100111111111000000;

scoremem[6] =
118'b001111111111110000000011111111000000000000000000111111111111110000000000111000011111111000000000011100111111111000000;

scoremem[7] =
118'b000111111111111000000011111111000000000000000000111111111111110000000000011100011111111000000000011100111111111000000;

scoremem[8] =
118'b000001111111111100000011111111000000000000000000111111111111110000000000011100011111111000000000011100111111111000000;

scoremem[9] =
118'b000000111111111111100001111111000000000000000001111111111111110000000000011100011111111011100000000000011111110000000;

scoremem[10] =
118'b000000001111111111110000111111000000000000000001111111111111110000000000011100011111111000111000000000011111110000000;

scoremem[11] =
118'b000000000111111111100000011111000000000000000001111111111111110000000000111000011111111000001110000000011111110000000;

scoremem[12] =
118'b000000000011111111000000000111000000000000000001111111111111110000000000111000011111111000001110000000011111110000000;

scoremem[13] =
118'b000000000001111000000000000001110000000001110000000111111111110000000111000000011111111000001110000000011111110000000;

scoremem[14] =
118'b111000011100000000000000000000001110011100000000000000111111100011100000000000011111111000001110000000011111111111111;

scoremem[15] =
118'b000111000000000000000000000000000001110000000000000000000011100000000000000011111111000001110000000011111111111111;

//TIME

timemem[0] =
102'b11111111111111111111100011111111111000000000011111100000000000000000011111100000000000111111111111110;

timemem[1] =
102'b11111111111111111111100011111111111000000000011111100000000000000000111111100000000000111111111111110;

timemem[2] =
102'b00000111111111111000000001111111111100000000011111111100000000000000111111111000000000111111111110000;

timemem[3] =
102'b00000111111111111000000001111111111100000000011111111100000000000011111111110000000000111111111110000;

timemem[4] =
102'b000001111111111100000000111111111110000000011111111111000000000111111111110000000000111111111110000000;

timemem[5] =
102'b000001111111111100000000111111111110000000011111111111110000011111111111110000000000111111111110000000;

timemem[6] =
102'b000001111111111100000000111111111110000000111001111111111101110001111111111000000000111111111110000000;

timemem[7] =
102'b000001111111111100000000111111111110000000111000111111111111000001111111111000000000111111111110000000;

timemem[8] =
102'b000001111111111100000000111111111110000001110000011111111111000000011111111111000000000111111111110000;

timemem[9] =
102'b000001111111111100000000111111111110000001110000001111111110000000011111111111000000000111111111110000;

timemem[10] =
102'b000001111111111100000000111111111110000011100000000111111100000000000111111111111000000000111111111110000;

timemem[11] =
102'b000001111111111100000000111111111110000011100000000001111100000000000001111111111100000000111111111110000;

timemem[12] =
102'b000001111111111100000000111111111110000111000000000001100000000000000001111111111100000000111111111110000;

timemem[13] =
102'b000001111111111100000000111111111110000111000000000000000000000000000000111111111100000000111111111110000;

timemem[14] =
102'b000001111111111100000000111111111110001110000000000000000000000000000000111111111100000000111111111111110;

timemem[15] =
102'b000001111111111100000000111111111110001110000000000000000000000000000000011111111100000000111111111111110;

//WHAC-A-MOLE

momem[0] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[1] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[2] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[3] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[4] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[5] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[6] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[7] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[8] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[9] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[10] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[11] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[12] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[13] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[14] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[15] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[16] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[17] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem[18] =
64'b1111100000000000000000000011111111110000000000000000000011111;

momem[19] =
64'b1111100000000000000000000111110011111000000000000000000011111;

momem[20] =
64'b1111100000000000000000000111110000111110000000000000000011111;

momem[21] =
64'b1111100000000000000000000111110000011111000000000000000011111;

momem[22] =
64'b1111100000000000000000000111110000000111110000000000000011111;

momem[23] =
64'b1111100000000000000000000111110000000011111000000000000011111;

momem[24] =
64'b1111100000000000000000000111110000000000111110000000000011111;

momem[25] =
64'b1111100000000000000000000111110000000000001111100000000011111;

momem[26] =
64'b1111100000000000000000000111110000000000000011111000000011111;

momem[27] =
64'b1111100000000000000000000111110000000000000000111110000011111;

momem[28] =
64'b1111100000000000000000000111110000000000000000011111000011111;

momem[29] =
64'b1111100000000000000000000111110000000000000000001111100011111;

momem[30] =
64'b1111100000000000000000000111110000000000000000000111110011111;

momem[31] =
64'b1111100000000000000000000111110000000000000000000011111011111;

```
        momem[32] =
64'b1111100000000111110000000000000000000000000001111100000000011111;

        momem[33] =
64'b1111100000001111100000000000000000000000000000111110000000011111;

        momem[34] =
64'b1111100000011111000000000000000000000000000000111111000000011111;

        momem[35] =
64'b1111100000111110000000000000000000000000000000011111000000011111;

        momem[36] =
64'b1111100001111100000000000000000000000000000000011111000011111;

        momem[37] =
64'b1111100011111000000000000000000000000000000000011111100011111;

        momem[38] =
64'b1111001111100000000000000000000000000000000000011111100111111;

        momem[39] =
64'b1111011111000000000000000000000000000000000000011111011111;

        momem[40] =
64'b1111111111000000000000000000000000000000000000001111111111;


          momem2[0]
=64'b1111100000000000000000000000000000000000000000000000000011111;

          momem2[1]
=64'b1111100000000000000000000000000000000000000000000000000011111;

          momem2[2]
=64'b1111100000000000000000000000000000000000000000000000000011111;

          momem2[3]
=64'b1111100000000000000000000000000000000000000000000000000011111;

          momem2[4]
=64'b1111100000000000000000000000000000000000000000000000000011111;

          momem2[5]
=64'b1111100000000000000000000000000000000000000000000000000011111;

          momem2[6]
=64'b1111100000000000000000000000000000000000000000000000000011111;

          momem2[7]
=64'b1111100000000000000000000000000000000000000000000000000011111;
```

momem2[8]
=64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[9] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[10] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[11] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[12] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[13] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[14] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[15] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[16] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[17] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[18] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[19] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem2[20] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem2[21] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem2[22] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem2[23] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem2[24] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[25] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[26] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[27] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[28] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[29] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[30] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[31] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[32] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[33] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[34] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[35] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[36] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[37] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[38] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[39] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem2[40] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[0]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[1]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[2]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[3]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[4]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[5]
=64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[6]
=64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[7]
=64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[8]
=64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[9] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[10] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[11] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[12] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[13] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[14] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[15] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[16] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[17] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

momem3[18] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[19] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[20] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[21] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[22] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[23] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momem3[24] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[25] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[26] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[27] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[28] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[29] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[30] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[31] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[32] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[33] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[34] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[35] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[36] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[37] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[38] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[39] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momem3[40] =
64'b1111100000000000000000000000000000000000000000000000000011111;

momemc[0]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[1]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[2]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[3]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[4]
=64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[5]
=64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[6]
=64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[7]
=64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[8]
=64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[9] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[10] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[11] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[12] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[13] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[14] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[15] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[16] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[17] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[18] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[19] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[20] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[21] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[22] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[23] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[24] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[25] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[26] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[27] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[28] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[29] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[30] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[31] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[32] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[33] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[34] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[35] =
64'b1111100000000000000000000000000000000000000000000000000000000000;

momemc[36] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[37] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[38] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[39] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

momemc[40] =
64'b1111111111111111111111111111111111111111111111111111111111111111;

mmem[0] =
64'b1111100000000000000000000000000000000000000000000000000000011111;

mmem[1] =
64'b1111110000000000000000000000000000000000000000000000000000111111;

mmem[2] =
64'b1111111000000000000000000000000000000000000000000000000001111111;

mmem[3] =
64'b1111111100000000000000000000000000000000000000000000000011111111;

mmem[4] =
64'b1111111110000000000000000000000000000000000000000000000111111111;

mmem[5] =
64'b1111111111000000000000000000000000000000000000000000001111111111;

mmem[6] =
64'b1111101111100000000000000000000000000000000000000000011111011111;

mmem[7] = 64'b1111100111110000000000000000000000000000000000000000111110011111;

mmem[8] = 64'b1111100011111000000000000000000000000000000000000001111100011111;

mmem[9] = 64'b1111100001111100000000000000000000000000000000000011111000011111;

mmem[10] = 64'b1111100000011111000000000000000000000000000000001111100000011111;

mmem[11] = 64'b1111100000001111110000000000000000000000000000011111100000011111;

mmem[12] = 64'b1111100000000011111100000000000000000000000001111110000000011111;

mmem[13] = 64'b1111100000000001111110000000000000000000000111111000000000011111;

mmem[14] = 64'b1111100000000000111111000000000000000000011111100000000000011111;

mmem[15] = 64'b1111100000000000001111110000000000000001111110000000000000011111;

mmem[16] = 64'b1111100000000000000011111100000000000111111000000000000000011111;

mmem[17] = 64'b1111100000000000000000111111000001111110000000000000000000011111;

mmem[18] = 64'b1111100000000000000000011111111111110000000000000000000000011111;

mmem[19] = 64'b1111100000000000000000001111111111100000000000000000000000011111;

mmem[20] = 64'b1111100000000000000000001111111111100000000000000000000000011111;

mmem[21] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

mmem[22] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

mmem[23] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

mmem[24] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

mmem[25] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

mmem[26] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

```
mmem[27] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[28] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[29] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[30] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[31] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[32] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[33] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[34] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[35] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[36] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[37] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[38] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[39] = 64'b1111100000000000000000000000000000000000000000000000000000011111;
mmem[40] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

omem[0] = 64'b0000000000000000000000000111111111100000000000000000000000000000;
 omem[1] = 64'b0000000000000000000000011111111111111111100000000000000000000000;
 omem[2] = 64'b0000000000000000000111111111111111111111111100000000000000000000;
 omem[3] = 64'b0000000000000001111111110000000000000001111111110000000000000000;
 omem[4] = 64'b0000000000001111111100000000000000000001111111100000000000000000;
 omem[5] = 64'b0000000000111111100000000000000000000000111111100000000000000000;
 omem[6] = 64'b0000000011111100000000000000000000000000001111110000000000000000;
 omem[7] = 64'b0000001111110000000000000000000000000000000011111100000000000000;
 omem[8] = 64'b0000011111000000000000000000000000000000000001111100000000000000;
 omem[9] = 64'b0000111110000000000000000000000000000000000000111110000000000000;
omem[10] = 64'b0000111110000000000000000000000000000000000000111110000000000000;
omem[11] = 64'b0001111100000000000000000000000000000000000000011111000000000000;
omem[12] = 64'b0011111000000000000000000000000000000000000000001111000000000000;
omem[13] = 64'b0011111000000000000000000000000000000000000000001111000000000000;
```

omem[14] = 64'b0011111000000000000000000000000000000000000000000000000001111100;

omem[15] = 64'b0011111000000000000000000000000000000000000000000000000001111100;

omem[16] = 64'b0111110000000000000000000000000000000000000000000000000000111110;

omem[17] = 64'b0111110000000000000000000000000000000000000000000000000000111110;

omem[18] = 64'b0111110000000000000000000000000000000000000000000000000000111110;

omem[19] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

omem[20] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

omem[21] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

omem[22] = 64'b1111100000000000000000000000000000000000000000000000000000011111;

omem[23] = 64'b0111110000000000000000000000000000000000000000000000000000111110;

omem[24] = 64'b0111110000000000000000000000000000000000000000000000000000111110;

omem[25] = 64'b0111110000000000000000000000000000000000000000000000000000111110;

omem[26] = 64'b0011111000000000000000000000000000000000000000000000000001111100;

omem[27] = 64'b0011111000000000000000000000000000000000000000000000000001111100;

omem[28] = 64'b0011111000000000000000000000000000000000000000000000000001111100;

omem[29] = 64'b0001111100000000000000000000000000000000000000000000000011111000;

omem[30] = 64'b0000111110000000000000000000000000000000000000000000000111110000;

omem[31] = 64'b0000011111000000000000000000000000000000000000000000001111100000;

omem[32] = 64'b0000001111100000000000000000000000000000000000000000011111000000;

omem[33] = 64'b0000000111111000000000000000000000000000000000000001111110000000;

omem[34] = 64'b0000000001111110000000000000000000000000000000000111111000000000;

omem[35] = 64'b0000000000011111110000000000000000000000000000011111110000000000;

omem[36] = 64'b0000000000000111111110000000000000000000000001111111100000000000;

omem[37] = 64'b0000000000000001111111110000000000000000011111111100000000000000;

omem[38] = 64'b0000000000000000001111111111111111111111111110000000000000000000;

omem[39] = 64'b0000000000000000000001111111111111111110000000000000000000000000;

omem[40] = 64'b0000000000000000000000001111111111000000000000000000000000000000;

lmem[0] = 64'b1111100000000000000000000000000000000000000000000000000000000000;

```
 lmem[1] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[2] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[3] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[4] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[5] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[6] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[7] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[8] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 lmem[9] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[10] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[11] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[12] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[13] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[14] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[15] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[16] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[17] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[18] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[19] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[20] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[21] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[22] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[23] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[24] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[25] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[26] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[27] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[28] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[29] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
```

```
lmem[30] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[31] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[32] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[33] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[34] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[35] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
lmem[36] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
lmem[37] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
lmem[38] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
lmem[39] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
lmem[40] = 64'b1111111111111111111111111111111111111111111111111111111111111111;


 emem[0] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
 emem[1] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
 emem[2] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
 emem[3] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
 emem[4] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
 emem[5] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 emem[6] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 emem[7] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 emem[8] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
 emem[9] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[10] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[11] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[12] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[13] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[14] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
```

```
emem[15] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[16] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[17] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[18] = 64'b1111111111111111111111111111111111000000000000000000000000000000;
emem[19] = 64'b1111111111111111111111111111111111000000000000000000000000000000;
emem[20] = 64'b1111111111111111111111111111111111000000000000000000000000000000;
emem[21] = 64'b1111111111111111111111111111111111000000000000000000000000000000;
emem[22] = 64'b1111111111111111111111111111111111000000000000000000000000000000;
emem[23] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[24] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[25] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[26] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[27] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[28] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[29] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[30] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[31] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[32] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[33] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[34] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[35] = 64'b1111100000000000000000000000000000000000000000000000000000000000;
emem[36] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
emem[37] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
emem[38] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
emem[39] = 64'b1111111111111111111111111111111111111111111111111111111111111111;
emem[40] = 64'b1111111111111111111111111111111111111111111111111111111111111111;

dash[0]= 16'b0000000000000000;
dash[1]= 16'b0000000000000000;
```

```
dash[2]= 16'b0000000000000000;

dash[3]= 16'b1111111111111111;

dash[4]= 16'b1111111111111111;

dash[5]= 16'b1111111111111111;

dash[6]= 16'b1111111111111111;

dash[7]= 16'b1111111111111111;

dash[8]= 16'b1111111111111111;

dash[9]= 16'b1111111111111111;

dash[10]=16'b1111111111111111;

dash[11]=16'b1111111111111111;

dash[12]=16'b1111111111111111;

dash[13]=16'b0000000000000000;

dash[14]=16'b0000000000000000;

dash[15]=16'b0000000000000000;


        //HAMMER

 hammermem[0] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

 hammermem[1] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

 hammermem[2] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

 hammermem[3] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

 hammermem[4] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
```

hammermem[5] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[6] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[7] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[8] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[9] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[10] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[11] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[12] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[13] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[14] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[15] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[16] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[17] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[18] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[19] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[20] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[21] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[22] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[23] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000011111111111110000000000000000;

hammermem[24] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000001111111111111111111110000000000;

hammermem[25] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000011111111111111111111111110000000;

hammermem[26] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111111111111110000;

hammermem[27] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111111111111111100;

hammermem[28] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111111111111111100;

hammermem[29] =
96'b000000000000000000000000000000000000000000011111111111111111111111111111111111111111111110;

hammermem[30] =
96'b000000000000000000000000000000000000000000111111111111111111111111111111111111111111111110;

hammermem[31] =
96'b000000000000000000000000000000000000000001111111111111111111111111111111111111111111111110;

hammermem[32] =
96'b000000000000000000000000000000000000000011111111111111111111111111111111111111111111111111;

hammermem[33] =
96'b000000000000000000000000000000111111111111111111111111111111111111111111111111111111111111;

hammermem[34] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111111111111111111111;

hammermem[35] =
96'b000000000000000000001111111111111111111111111111111111111111111111111111111111111111111111;

hammermem[36] =
96'b000000000000000111111111111111111111111111111111111111111111111111111111111111111111111111;

hammermem[37] =
96'b000000000001111111111111111111111111111111111111111111111111111111111111111111111111111110;

hammermem[38] =
96'b000000011111111111111111111111111111111111111111111111111111111111111111111111111111111100;

hammermem[39] =
96'b000001111111111111111111111111111111111111111111111111111111111111111111111111111111111100;

hammermem[40] =
96'b000011111111111111111111111111111111111111111111111111111111111111111111111111111111111000;

hammermem[41] =
96'b000111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111100000;

hammermem[42] =
96'b001111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111100000000;

hammermem[43] =
96'b011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111100000000000;

hammermem[44] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1110000000000000;

hammermem[45] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111111111111111110
0000000000000000;

hammermem[46] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111111111111000000
0000000000000000;

hammermem[47] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111111111000000000
0000000000000000;

hammermem[48] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111111100000000000
0000000000000000;

hammermem[49] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111110000000000000
0000000000000000;

hammermem[50] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111100000000000000
0000000000000000;

hammermem[51] =
96'b111111111111111111111111111111111111111111111111111111111111111111111111111000000000000000
0000000000000000;

hammermem[52] =
96'b011111111111111111111111111111111111111111111111111111111111111111111111111000000000000000
0000000000000000;

hammermem[53] =
96'b011111111111111111111111111111111111111111111111111101111111110000000000000000000000000000000000;

hammermem[54] =
96'b001111111111111111111111111111111111111111111100000000111111111100000000000000000000000000000000;

hammermem[55] =
96'b000011111111111111111111111111111111111111111100000000000011111111110000000000000000000000000000;

hammermem[56] =
96'b000001111111111111111111111111111111111111110000000000000001111111110000000000000000000000000000;

hammermem[57] =
96'b000000011111111111111111111111111111111110000000000000000011111111100000000000000000000000000000;

hammermem[58] =
96'b000000001111111111111111111111111111100000000000000000000001111111110000000000000000000000000000;

hammermem[59] =
96'b000000000001111111111111111111110000000000000000000000000001111111110000000000000000000000000000;

hammermem[60] =
96'b000000000000000011111111111000000000000000000000000000000001111111110000000000000000000000000000;

hammermem[61] =
96'b000000000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000;

hammermem[62] =
96'b000000000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000;

hammermem[63] =
96'b000000000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000;

hammermem[64] =
96'b000000000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000;

hammermem[65] =
96'b000000000000000000000000000000000000000000000000000000000000000000000111111111100000000000000000000000;

hammermem[66] =
96'b000000000000000000000000000000000000000000000000000000000000000000000011111111110000000000000000000000;

hammermem[67] =
96'b000000000000000000000000000000000000000000000000000000000000000000000001111111111000000000000000000000;

hammermem[68] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000111111111100000000000000000000;

hammermem[69] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000011111111110000000000000000000;

hammermem[70] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000001111111111000000000000000000;

hammermem[71] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111100000000000000000;

hammermem[72] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000011111111110000000000000000;

hammermem[73] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000001111111111000000000000000;

hammermem[74] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000111111111100000000000000;

hammermem[75] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000011111111110000000000000;

hammermem[76] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111000000000000;

hammermem[77] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111000000000;

hammermem[78] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111100000000;

hammermem[79] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111110000000;

hammermem[80] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111000000;

hammermem[81] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111100000;

hammermem[82] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111110000;

hammermem[83] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111000;

hammermem[84] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111100;

hammermem[85] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111110;

hammermem[86] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111;

hammermem[87] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111;

hammermem[88] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111;

hammermem[89] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111;

hammermem[90] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111;

hammermem[91] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111;

hammermem[92] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001110;

hammermem[93] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[94] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammermem[95] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[0] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[1] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[2] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[3] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[4] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[5] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[6] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[7] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[8] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[9] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[10] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[11] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[12] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[13] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[14] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[15] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[16] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[17] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[18] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[19] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[20] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[21] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

hammer2mem[22] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000011111111111110000000000000000;

hammer2mem[23] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111100000000000;

hammer2mem[24] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111110000000000;

hammer2mem[25] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111110000000000;

hammer2mem[26] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111111000000000;

hammer2mem[27] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111111100000000;

hammer2mem[28] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111111110000000;

hammer2mem[29] =
96'b000000000000000000000000000000000000000000000000000000000000000001111111111111111
1111111111111111;

hammer2mem[30] =
96'b000000000000000000000000000000000000000000000000000000000000000001111111111111111
1111111111111111;

hammer2mem[31] =
96'b000000000000000000000000000000000000000000000000000000000000000001111111111111111
1111111111111111;

hammer2mem[32] =
96'b000000000000000000000000000000000000000000000000000000000000000001111111111111111
1111111111111111;

hammer2mem[33] =
96'b000000000000000000000000000000000000000000000000000000000000000011111111111111111
1111111111111111;

hammer2mem[34] =
96'b000000000000000000000000000000000000000000000000000000000000000011111111111111111
1111111111111111;

hammer2mem[35] =
96'b000000000000000000000000000000000000000000000000000000000000000111111111111111111
1111111111111111;

hammer2mem[36] =
96'b000000000000000000000000000000000000000000000000000000000000000111111111111111111
1111111111111111;

hammer2mem[37] =
96'b000000000000000000000000000000000000000000000000000000000000001111111111111111111
1111111111111111;

hammer2mem[38] =
96'b000000000000000000000000000000000000000000000000000000000000001111111111111111111
1111111111111110;

hammer2mem[39] =
96'b000000100000000000000000000000000000000000000000000000000000000011111111111111111
1111111111111110;

hammer2mem[40] =
96'b000011100000000000000000000000000000000000000000000000000000000000111111111111111
1111111111111110;

hammer2mem[41] =
96'b000111000000000000000000000000000000000000000000000000000000000000000000000000000000000011
1111111111111000;

hammer2mem[42] =
96'b001110000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0001111111100000;

hammer2mem[43] =
96'b011110000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer2mem[44] =
96'b111110000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer2mem[45] =
96'b111100000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer2mem[46] =
96'b111100000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer2mem[47] =
96'b111100000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer2mem[48] =
96'b111100000000000000000000000000000000000000000000000011111111000000000000000000000000000000
0000000000000000;

hammer2mem[49] =
96'b111100000000000000000000000000000000000000000000000111111111100000000000000000000000000000
0000000000000000;

hammer2mem[50] =
96'b111110000000000000000000000000000000000000000000001111111111110000000000000000000000000000
0000000000000000;

hammer2mem[51] =
96'b111110000000000000000000000000000000000000000000001111111111110000000000000000000000000000
0000000000000000;

hammer2mem[52] =
96'b011110000000000000000000000000000000000000000000001111111111110000000000000000000000000000
0000000000000000;

hammer2mem[53] =
96'b011111100000000000000000000000000000000000000000000011111111100000000000000000000000000000000000;

hammer2mem[54] =
96'b001111110000000000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[55] =
96'b000011111100000000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[56] =
96'b000001111111000000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[57] =
96'b000000111111110000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[58] =
96'b000000011111111111000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[59] =
96'b000000000011111111111111000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[60] =
96'b000000000000000011111111111000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[61] =
96'b000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[62] =
96'b000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[63] =
96'b000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[64] =
96'b000000000000000000000000000000000000000000000000000001111111110000000000000000000000000000000000;

hammer2mem[65] =
96'b000000000000000000000000000000000000000000000000000000000000000000000111111111000000000000000000000;

hammer2mem[66] =
96'b000000000000000000000000000000000000000000000000000000000000000000000011111111100000000000000000000;

hammer2mem[67] =
96'b000000000000000000000000000000000000000000000000000000000000000000000001111111110000000000000000000;

hammer2mem[68] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000111111111000000000000000000;

hammer2mem[69] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000011111111100000000000000000;

hammer2mem[70] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000001111111110000000000000000;

hammer2mem[71] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000111111111000000000000000;

hammer2mem[72] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000011111111100000000000000;

hammer2mem[73] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000001111111110000000000000;

hammer2mem[74] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000111111111000000000000;

hammer2mem[75] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000011111111100000000000;

hammer2mem[76] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000001111111110000000000;

hammer2mem[77] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111000000000;

hammer2mem[78] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111100000000;

hammer2mem[79] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111110000000;

hammer2mem[80] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111000000;

hammer2mem[81] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111100000;

hammer2mem[82] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111110000;

hammer2mem[83] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111000;

hammer2mem[84] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111100;

hammer2mem[85] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111110;

hammer2mem[86] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111;

hammer2mem[87] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111;

hammer2mem[88] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111;

hammer2mem[89] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000001111111;

hammer2mem[90] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000111111;

hammer2mem[91] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000011111;

hammer2mem[92] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000001110;

hammer2mem[93] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer2mem[94] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer2mem[95] =
96'b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000;

hammer3mem[0] =
96'b000000000000000000001000000000000000010000000000000001100000000000000000100000
0000000000010000000000000;

hammer3mem[1] =
96'b000000000000000000001100000000000000011000000000000001111000000000000001100000
0000000000110000000000000;

hammer3mem[2] =
96'b000000000000000000001110000000000000111000000000000011111100000000000011100000
0000000011100000000000000;

hammer3mem[3] =
96'b000000000000000000001111000000000000111100000000000111111110000000000111100000
0000001111000000000000000;

hammer3mem[4] =
96'b000000000000000000000000011111000000000111110000000011111111110000000011111000000000111111000000000000000000;

hammer3mem[5] =
96'b000000000000000000000000011111100000001111110000000111111111111000000111111000000011111110000000000000000000;

hammer3mem[6] =
96'b000000000000000000000000011111110000011111110000011111111111111000011111110000001111111000000000000000000000;

hammer3mem[7] =
96'b000000000000000000000000011111111000111111110011111111111111111001111111100011111111000000000000000000000000;

hammer3mem[8] =
96'b000000000000000000000000011111111101111111111111111111111111111111111101111111100000000000000000000000000;

hammer3mem[9] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111111111100000000000000000000000000;

hammer3mem[10] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111111111000000000000000000000000000;

hammer3mem[11] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111111110000000000000000000000000000;

hammer3mem[12] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111111100000000000000000000000000000;

hammer3mem[13] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111111000000000000000000000000000000;

hammer3mem[14] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111110000000000000000000000000000000;

hammer3mem[15] =
96'b000000000000000000000000011111111111111111111111111111111111111111111111100000000000000000000000000000000;

hammer3mem[16] =
96'b000000000000000000000000000000000000011111111111111111111111111111111111111000
000000000000000000000000;

hammer3mem[17] =
96'b000000000000000000000000000000000000011111111111111111111111111111111111110000
000000000000000000000000;

hammer3mem[18] =
96'b000000000000000000000000000000000000011111111111111111111111111111111111100000
000000000000000000000000;

hammer3mem[19] =
96'b000000000000000000000000000000000000011111111111111111111111111111111111000000
000000000000000000000000;

hammer3mem[20] =
96'b000000000000000000000000000000000000011111111111111111111111111111111110000000
000000000000000000000000;

hammer3mem[21] =
96'b000000000000000000000000000000000000011111111111111111111111111111111100000000
000000000000000000000000;

hammer3mem[22] =
96'b000000000000000000000000000000000000011111111111111111111111111111111000000000
000000000000000000000000;

hammer3mem[23] =
96'b000000000000000000000000000000000000011111111111111111111111111111110000000000
000000000000000000000000;

hammer3mem[24] =
96'b000000000000000000000000000000000000011111111111111111111111111111100000000000
000000000000000000000000;

hammer3mem[25] =
96'b000000000000000000000000000000000000011111111111111111111111111111000000000000
000000000000000000000000;

hammer3mem[26] =
96'b000000000000000000000000000000000000011111111111111111111111111110000000000000
000000000000000000000000;

hammer3mem[27] =
96'b000000000000000000000000000000000000011111111111111111111111111100000000000000
000000000000000000000000;

```
hammer3mem[28] =
96'b000000000000000000000000000000000000000000000001111111111111110000000000000000000000000000000000000000;

hammer3mem[29] =
96'b000000000000000000000000000000000000000000000001111111111111110000000000000000000000000000000000000000;

hammer3mem[30] =
96'b000000000000000000000000000000000000000000000001111111111111000000000000000000000000000000000000000000;

hammer3mem[31] =
96'b000000000000000000000000000000000000000000000001111111111100000000000000000000000000000000000000000000;


 /*rockmem[0] = 20'b00000000111000000000;

 rockmem[1] = 20'b00000001111110000000;

 rockmem[2] = 20'b00001111111111110000;

 rockmem[3] = 20'b00011111111111111000;

 rockmem[4] = 20'b00111111111111111100;

 rockmem[5] = 20'b00111111111111111100;

 rockmem[6] = 20'b00011111111111111000;

 rockmem[7] = 20'b00011111111111111000;

 rockmem[8] = 20'b00111111111111111100;

 rockmem[9] = 20'b00111111111111111100;

rockmem[10] = 20'b01111111111111111110;

rockmem[11] = 20'b01111111111111111110;

rockmem[12] = 20'b01111111111111111110;

rockmem[13] = 20'b00111111111111111100;

rockmem[14] = 20'b00111111111111111100;

rockmem[15] = 20'b01111111111111111110;

rockmem[16] = 20'b00111111111111111100;

rockmem[17] = 20'b00111111111111111100;

rockmem[18] = 20'b00001111111111110000;
```

```
rockmem[19] = 20'b00000000111100000000;


 rockmem2[0] = 20'b00000001111100000000;
 rockmem2[1] = 20'b00000011000111000000;
 rockmem2[2] = 20'b00011110000001111000;
 rockmem2[3] = 20'b00110000000000001100;
 rockmem2[4] = 20'b01100000000000000110;
 rockmem2[5] = 20'b01100000000000000010;
 rockmem2[6] = 20'b00100000000000000110;
 rockmem2[7] = 20'b01100000000000000100;
 rockmem2[8] = 20'b01000000000000000110;
 rockmem2[9] = 20'b01000000000000000010;
rockmem2[10] = 20'b11000000000000000011;
rockmem2[11] = 20'b10000000000000000001;
rockmem2[12] = 20'b11000000000000000011;
rockmem2[13] = 20'b01000000000000000010;
rockmem2[14] = 20'b01000000000000000010;
rockmem2[15] = 20'b10000000000000000011;
rockmem2[16] = 20'b00000000000000000010;
rockmem2[17] = 20'b01100000000000001110;
rockmem2[18] = 20'b00011111000011111000;
rockmem2[19] = 20'b00000001111110000000;*/


////////////////////////////////////////stones////////////////////////////////////////////////////////////////
/////


                                stonemem[0] =
120'b000000000000000000000000000000000000000000000000000000000000011100000000000000000000
00000000000000000000000000000000000000000000;
```

stonemem[1] =
120'b000000000000000000000000000000000000000000011110000000011111111110000000000111110
000000000000000000000000000000000000000000;

stonemem[2] =
120'b000000000000000000000000000000000000000001111111111000011111111111110000001111111
111000000000000110000000000000000000000000;

stonemem[3] =
120'b000000000000000000000000000000000000011111111111111111111111111111111111111111111
111111110000111111111000000000000000000000;

stonemem[4] =
120'b000000000000000000000000000000000000111111111111111111111111111111111111111111111
111111111111111111111100000000000000000000;

stonemem[5] =
120'b000000000000000000000000000000000011111111111111111111111111111111111111111111111
111111111111111111111110000000000000000000;

stonemem[6] =
120'b000000000000000000000000000000011111111111111111111111111111111111111111111111111
111111111111111111111110000000000000000000;

stonemem[7] =
120'b000000000000000000000000000001111111111111111111111111111111111111111111111111111
111111111111111111111110000000000000000000;

stonemem[8] =
120'b000000000000000000011111111111111111111111111111111111111111111111111111111111111
111111111111111111111110000000000000000000;

stonemem[9] =
120'b000000000000000001111111111111111111111111111111111111111111111111111111111111111
111111111111111111111110000000000000000000;

stonemem[10] =
120'b000000000000000001111111111111111111111111111111111111111111111111111111111000011
111111111111111111111110000000000000000000;

stonemem[11] =
120'b000000000000000001111111111111111111000000111111111011111100000111111111000000000
111111111111111111111001000000000000000000;

stonemem[12] =
120'b000000000000000001111111111111111110000001111111100001111100000001111110000000000
000111110000000111111111111111100000000000;

stonemem[13] =
120'b000000000000000001111111000000011100000011111110000001111000000011111100000000
001111100000000001111111111100000000000;

stonemem[14] =
120'b000000000000000011111111111000000000000000011111000000011100000000111100000000000
001111100000000000011111111000000000000;

stonemem[15] =
120'b000000000000000011111111110000000000000000011111000000001100000000011000000000000
001111100000000000011111111000000000000;

stonemem[16] =
120'b000000000000000011111111111110000000000000001111100000000000000000000000000000000
000111100000000001111111110000000000000;

stonemem[17] =
120'b000000000001111111111111111100000000000000001110000000000000000000000000000000000
000000000000000000000011111100000000000;

stonemem[18] =
120'b000000000001111111111111111100000000000000011100000000000000000000000000000000000
000000000000000000000111000000000000000;

stonemem[19] =
120'b000000000001111111111111111100000000000000000000000000000000000000000000000000000
000000000000000000001111111111000000000;

stonemem[20] =
120'b000000000011111111000000000000000000000000000000000000000000000000000000000000000
000000000000000000001111111111000000000;

stonemem[21] =
120'b000000001111111111000000000000000000000000000000000000000000000000000000000000000
000000000000000000001111111110000000000;

stonemem[22] =
120'b000000111111111111000000000000000000000000000000000000000000000000000000000000000
000000000000000000001111111110000000000;

stonemem[23] =
120'b000000111111111111000000000000000000000000000000000000000000000000000000000000000
000000000000000000001111111111000000000;

stonemem[24] =
120'b000111111111111111100000000000000000000000000000000000000000000000000000000000000
000000000000000000001111111111110000000;

stonemem[25] =
120'b0000011111111111111000000000000000000000000000000000000000000000000000000000000
00000000000111111111111111111111111110000;

stonemem[26] =
120'b0000001111111111111000000000000000000000000000000000000000000000000000000000000
00000000011111111111111111111111111100000;

stonemem[27] =
120'b0000011111111111110000011100000000000000000000000000000000000000000000000000000
00000000111111111111111111111111110000000;

stonemem[28] =
120'b0000011111111111111000111111100000000000000000000000000000000000000000000000000
00000000111111111111111111111111100000000;

stonemem[29] =
120'b0000001111111111111111111111000000000000000000000000000000000000000000000000000
00000000001111111111111111111111100000000;

stonemem[30] =
120'b0000000011111111111101111111000000000000000000000000000000000000000000000000000
00000000000011111111111111111111110000000;

stonemem[31] =
120'b1111111111111111111111111111100000000000000000000000000000000000000000000000000
00000000000000011111111111111111100000000;

stonemem[32] =
120'b1111111111111111111111111111000000000000000000000000000000000000000000000000000
00000000000000001111111111111111000000000;

stonemem[33] =
120'b1111111111111111111111111110000000000000000000000000000000000000000000000000000
00000000000000000011111111111100000000000;

stonemem[34] =
120'b1111111111111111110000000000000000000000000000000000000000000000000000000000000
00000000011111111111111110000000000000000;

stonemem[35] =
120'b1111111111111111111110000000000000000000000000000000000000000000000000000000000
00000000111111111111111110000000000000000;

stonemem[36] =
120'b1110011111111111111110000000000000000000000000000000000000000000000000000000000
00000001111111111111111111111111100000000;

stonemem[37] =
120'b0000001111111111111110000000000000000000000000000000000000000000000000000000000
000000011111111111111111111111111111000000;

stonemem[38] =
120'b0001111111111111111111100000000000000000000000000000000000000000000000000000000
000000011111111111111111111111111111110000;

stonemem[39] =
120'b0000011111111111111111000000011111111100000000000111111100000000000000000000000
0000000001111111111111111111111111111110000;

stonemem[40] =
120'b0000001111111111111111111111111111111111000000111111111111111111111111111111111
1111111111111111111111111111111111100000;

stonemem[41] =
120'b0000011111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111110000000;

stonemem[42] =
120'b0000011111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111100001111100000000;

stonemem[43] =
120'b0000001111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111000000000000000;

stonemem[44] =
120'b0000000011111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111000000000000000;

stonemem[45] =
120'b0000000000011111111100011111111111111111111111111111111111111111111111111111111
11111111111111111111111100000000000000000;

stonemem[46] =
120'b0000000000000111111000000111000111111111111111111111111111111111111111111111111
11111111111111111111111100000000000000000;

stonemem[47] =
120'b0000000000000000000000000000000011111111111111111111111111111111111111111111111
1111111111111111111111100000000000000000;

stonemem[48] =
120'b0000000000000000000000000000000001111111111111111111111111111111111111111111111
111111111111111111111100000000000000000;

```
                stonemem[49] =
120'b000000000000000000000000000000000011111111111111111111111111111111111111111
111111111111111111111110000000000000000;

                stonemem[50] =
120'b000000000000000000000000000000000000000000000001111111111111111111111000001 1
111111110000000000000000000000000000000;

                stonemem[51] =
120'b000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000;


                ////////////////////////////////grass
//////////////////////////////////////////////////////////////////////////

                grassmem[0] = 8'b00000000;

                grassmem[1] = 8'b00100000;

                grassmem[2] = 8'b01100000;

                grassmem[3] = 8'b01100100;

                grassmem[4] = 8'b11100110;

                grassmem[5] = 8'b11110110;

                grassmem[6] = 8'b11111111;

                grassmem[7] = 8'b11111111;


                 grass1mem[0] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

        grass1mem[1] =
64'b0000000000000000000000000000000110000000000000000000000000000000;

        grass1mem[2] =
64'b0000000000000000000000000000001111000000000000000000000000000000;

        grass1mem[3] =
64'b0000000000000000000000000000011111000000000000000000000000000000;

        grass1mem[4] =
64'b0000000000000000000000000000011111100000000000000000000000000000;

        grass1mem[5] =
64'b0000000000000000000000000000111111100000000000000000000000000000;
```

grass1mem[6] =
64'b1100000000000000000000000000000000011111111000000000000000000000;

grass1mem[7] =
64'b0111000000000000000000000000000000011111111000000000000000000000;

grass1mem[8] =
64'b0011110000000000000000000000000000011111111000000000000000000000;

grass1mem[9] =
64'b0001111000000000000000000000000000011111111000000000000000000000;

grass1mem[10] =
64'b0001111100000000000000000000000000011111111100000000000000000000;

grass1mem[11] =
64'b0000111110000000000000000000000011000111111111100000000000000000;

grass1mem[12] =
64'b0000111111000000000000000000011110000111111111100000000000000000;

grass1mem[13] =
64'b0000011111100000000000000000111110000111111111110000000000000000;

grass1mem[14] =
64'b0000011111110000000000000011111100000111111111110000000000000111;

grass1mem[15] =
64'b0000001111111000000000011111111000000111111111110000000000011110;

grass1mem[16] =
64'b0000001111111000000000111111110000000111111111110000000001111100;

grass1mem[17] =
64'b0000000111111100000001111111110000000111111111110000000111111000;

grass1mem[18] =
64'b0000000111111110000011111111110000000111111111110000011111110000;

grass1mem[19] =
64'b0000000011111111000111111111110000000111111111110001111111100000;

grass1mem[20] =
64'b0000000011111111000111111111110000001111111111110011111111000000;

grass1mem[21] =
64'b0000000001111111100111111111110000001111111111110111111110000000;

grass1mem[22] =
64'b0000000001111111100111111111110000011111111111111111111100000000;

grass1mem[23] =
64'b0000000000111111110011111111111100001111111111111111111000000000;

grass1mem[24] =
64'b0000000000111111111011111111111110011111111111111111110000000000;

grass1mem[25] =
64'b0000000000011111111011111111111110011111111111111111100000000000;

grass1mem[26] =
64'b0000000000011111111111111111111111011111111111111111100000000000;

grass1mem[27] =
64'b0000000000001111111111111111111111111111111111111111100000000000;

grass1mem[28] =
64'b0000000000011111111111111111111111111111111111111111110000000000;

grass1mem[29] =
64'b0000000000011111111111111111111111111111111111111111110000000000;

grass2mem[0] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[1] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[2] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[3] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[4] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[5] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[6] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[7] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[8] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[9] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[10] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[11] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[12] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[13] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[14] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[15] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[16] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[17] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[18] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[19] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[20] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[21] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[22] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[23] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[24] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[25] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[26] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[27] =
64'b0000000000000000000000000000000000000000000000000000000000000000;

grass2mem[28] =
64'b0000011100000000000001111111110000000000000000000000000000000000;

grass2mem[29] =
64'b0000111100000000001111111111111100000000000011111111111100000000;

grass2mem[30] =
64'b0001111110000000111111111111111111000000011111111111111111100000;

grass2mem[31] =
64'b0011111110000011111111111111111111100001111111111111111111111000;

grass2mem[32] =
64'b0111111110000111111111111111111111100111111111111111111111111100;

grass2mem[33] =
64'b1111111111001111111111111111111111111111111111111111111111111110;

grass2mem[34] =
64'b1111111111111111111111111111111111111111111111111111111111111110;

grass2mem[35] =
64'b1111111111111111111111111111111111100111111111111111111111111110;

grass2mem[36] =
64'b0111111110011111111111111111111111000011111111111111111111111100;

grass2mem[37] =
64'b0111111000001111111111111111111110000001111111111111111111111000;

grass2mem[38] =
64'b0000000000000111111111111111111000000000001111111111111110000000;

grass2mem[39] =
64'b0000000000000000111111111110000000000000000000000000000000000000;

//////////////////////////////////////score//
//////////////////////////

scoreboardmem[0] =
160'b0000000000011111111111111111111111111111111111111000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboardmem[1] =
160'b0000001111111111111111111111111111111111111111111111111111111111111100000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboardmem[2] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111110000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboardmem[3] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111000000000000000000000000000000000000000000000000000000000000000000;

scoreboardmem[4] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111000000000000000000000000000000000000000000000000000;

scoreboardmem[5] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111100000000000000000000000000000000000000000000000000;

scoreboardmem[6] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111110000000000000000000000000000000000000000000000000;

scoreboardmem[7] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000000000000000000000000000000000000000;

scoreboardmem[8] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000000000000000000000000000000000000000;

scoreboardmem[9] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000000000000000000000000000000000000000;

scoreboardmem[10] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000000000000000000000000000000000000000;

scoreboardmem[11] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000000000000000000000000000000000000000;

scoreboardmem[12] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[13] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[14] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[15] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[16] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[17] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[18] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[19] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[20] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[21] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[22] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[23] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111000000000000000000000000000;

scoreboardmem[24] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111110000000000000;

scoreboardmem[25] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111000000000000000;

scoreboardmem[26] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111100000000000000;

scoreboardmem[27] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111110000000000000;

scoreboardmem[28] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111000000000000;

scoreboardmem[29] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000;

scoreboardmem[30] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111110000000000;

scoreboardmem[31] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111000000000;

scoreboardmem[32] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111100000000;

scoreboardmem[33] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111110000000;

scoreboardmem[34] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111000000;

scoreboardmem[35] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111100000;

scoreboardmem[36] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111110;

scoreboardmem[37] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboardmem[38] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboardmem[39] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111110;

scoreboardmem[40] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111100;

scoreboardmem[41] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111000;

scoreboardmem[42] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111110000;

scoreboardmem[43] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111100000;

scoreboardmem[44] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111000000;

scoreboardmem[45] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111110000000;

scoreboardmem[46] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111100000000;

scoreboardmem[47] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111000000000;

scoreboardmem[48] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111110000000000;

scoreboardmem[49] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111100000000000;

scoreboardmem[50] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111000000000000;

scoreboardmem[51] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111110000000000000;

scoreboardmem[52] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111100000000000000;

scoreboardmem[53] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111000000000000000;

scoreboardmem[54] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111110000000000000000;

scoreboardmem[55] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111100000000000000000;

scoreboardmem[56] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111000000000000000000;

scoreboardmem[57] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111110000000000000000000;

scoreboardmem[58] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboardmem[59] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111000000000000000000000;

```
scoreboardmem[60] =
160'b00000000111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[61] =
160'b00000000111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[62] =
160'b00000000111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[63] =
160'b00000000111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[64] =
160'b00000000011111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[65] =
160'b00000000011111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[66] =
160'b00000000011111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[67] =
160'b00000000011111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[68] =
160'b00000000011111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[69] =
160'b00000000001111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[70] =
160'b00000000001111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboardmem[71] =
160'b00000000001111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111100000000000000000000000000;
```

scoreboardmem[72] =
160'b0000000000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111100000000000000000000000000000000000000000000000000000;

scoreboardmem[73] =
160'b0000000000000011111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboardmem[74] =
160'b0000000000000000000011111111111111111111111111111111111111110000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboard1mem[0] =
160'b0000000011111111111111111111111111111111111111111100000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboard1mem[1] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111100000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboard1mem[2] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;

scoreboard1mem[3] =
160'b0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111110000000000000000000000000000000000000000000000000000000000000000000000;

scoreboard1mem[4] =
160'b1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111100000000000000000000000000000000000000000000000000000000;

scoreboard1mem[5] =
160'b1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000000000000000000000000000000000000000000000;

scoreboard1mem[6] =
160'b1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111110000000000000000000000000000000000000000000000000000;

scoreboard1mem[7] =
160'b1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[8] =
160'b1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[9] =
160'b0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[10] =
160'b0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[11] =
160'b0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[12] =
160'b0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[13] =
160'b0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[14] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[15] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[16] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[17] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[18] =
160'b0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111000000000000000000000000000000000;

scoreboard1mem[19] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111110000000000000000000;

scoreboard1mem[20] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[21] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111000000000000000000000;

scoreboard1mem[22] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[23] =
160'b0001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111000000000000000000000;

scoreboard1mem[24] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111000000000000000000000;

scoreboard1mem[25] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111000000000000000000000;

scoreboard1mem[26] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[27] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[28] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[29] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[30] =
160'b0000111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[31] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111000000;

scoreboard1mem[32] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111100000;

scoreboard1mem[33] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111110000;

scoreboard1mem[34] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111000;

scoreboard1mem[35] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111100;

scoreboard1mem[36] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111110;

scoreboard1mem[37] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboard1mem[38] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboard1mem[39] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboard1mem[40] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboard1mem[41] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboard1mem[42] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboard1mem[43] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111;

scoreboard1mem[44] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111110;

scoreboard1mem[45] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111100;

scoreboard1mem[46] =
160'b0000011111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111000;

scoreboard1mem[47] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111110000;

scoreboard1mem[48] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111100000;

scoreboard1mem[49] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111000000;

scoreboard1mem[50] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111110000000;

scoreboard1mem[51] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111100000000;

scoreboard1mem[52] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111000000000;

scoreboard1mem[53] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111110000000000;

scoreboard1mem[54] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111100000000000;

scoreboard1mem[55] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111000000000000;

scoreboard1mem[56] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111110000000000000;

scoreboard1mem[57] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111100000000000000;

scoreboard1mem[58] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111000000000000000;

scoreboard1mem[59] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111110000000000000000;

scoreboard1mem[60] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111100000000000000000;

scoreboard1mem[61] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111000000000000000000;

scoreboard1mem[62] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111110000000000000000000;

scoreboard1mem[63] =
160'b0000001111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111100000000000000000000;

scoreboard1mem[64] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111000000000000000000000;

scoreboard1mem[65] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111110000000000000000000000;

scoreboard1mem[66] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111100000000000000000000000;

scoreboard1mem[67] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[68] =
160'b0000000111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[69] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[70] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[71] =
160'b0000000011111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[72] =
160'b0000000001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[73] =
160'b0000000001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[74] =
160'b0000000001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[75] =
160'b0000000001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[76] =
160'b0000000001111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111100000000000000000000000000;

scoreboard1mem[77] =
160'b0000000000011111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111100000000000000000000000000000000000000000000000000000000000000000;

scoreboard1mem[78] =
160'b0000000000000000111111111111111111111111111111111111111111111111111111000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000;

```
//NUMBERS
//ZERO

 zero[0] = 32'b00000000000000011000000000000000;
 zero[1] = 32'b00000000000111110001100000000000;
 zero[2] = 32'b00000000111111110000001100000000;
 zero[3] = 32'b00000011111111110000000011000000;
 zero[4] = 32'b00000111111111110000000001100000;
 zero[5] = 32'b00001111111111110000000000110000;
 zero[6] = 32'b00011111111111110000000000011000;
 zero[7] = 32'b00111111111111110000000000001100;
 zero[8] = 32'b01111111111111110000000000000110;
 zero[9] = 32'b11111111111111110000000000000011;
zero[10] = 32'b11111111111111110000000000000011;
zero[11] = 32'b11111111111111110000000000000011;
zero[12] = 32'b11111111111111110000000000000011;
zero[13] = 32'b11111111111111110000000000000011;
zero[14] = 32'b11111111111111110000000000000011;
zero[15] = 32'b11111111111111110000000000000001;
zero[16] = 32'b11111111111111110000000000000011;
zero[17] = 32'b11111111111111110000000000000011;
zero[18] = 32'b11111111111111110000000000000011;
zero[19] = 32'b11111111111111110000000000000011;
zero[20] = 32'b11111111111111110000000000000011;
zero[21] = 32'b11111111111111110000000000000011;
zero[22] = 32'b01111111111111110000000000000110;
zero[23] = 32'b00111111111111110000000000001100;
zero[24] = 32'b00011111111111110000000000011000;
zero[25] = 32'b00001111111111110000000000110000;
```

```
 zero[26] = 32'b00000111111111110000000001100000;
 zero[27] = 32'b00000011111111110000000011000000;
 zero[28] = 32'b00000001111111110000000110000000;
 zero[29] = 32'b00000000011111110000001100000000;
 zero[30] = 32'b00000000000011110001100000000000;
 zero[31] = 32'b00000000000000011000000000000000;

//ONE
one[0] = 32'b00000000000000000000000000001111;
one[1] = 32'b00000000000000000001111111111111;
one[2] = 32'b00000000000111111111111111111111;
one[3] = 32'b00000011111111111111111111111111;
one[4] = 32'b00011111111111111111111111111111;
one[5] = 32'b11111000111111111111111111111111;
one[6] = 32'b00000000111111111111111111111111;
one[7] = 32'b00000000111111111111111111111111;
one[8] = 32'b00000000111111111111111111111111;
one[9] = 32'b00000000111111111111111111111111;
one[10] = 32'b00000000111111111111111111111111;
one[11] = 32'b00000000111111111111111111111111;
one[12] = 32'b00000000111111111111111111111111;
one[13] = 32'b00000000111111111111111111111111;
one[14] = 32'b00000000111111111111111111111111;
one[15] = 32'b00000000111111111111111111111111;
one[16] = 32'b00000000111111111111111111111111;
one[17] = 32'b00000000111111111111111111111111;
one[18] = 32'b00000000111111111111111111111111;
one[19] = 32'b00000000111111111111111111111111;
one[20] = 32'b00000000111111111111111111111111;
```

```
one[21] = 32'b00000000111111111111111111111111;
one[22] = 32'b00000000111111111111111111111111;
one[23] = 32'b00000000111111111111111111111111;
one[24] = 32'b00000000111111111111111111111111;
one[25] = 32'b00000000111111111111111111111111;
one[26] = 32'b00000000111111111111111111111111;
one[27] = 32'b00000000111111111111111111111111;
one[28] = 32'b00000000111111111111111111111111;
one[29] = 32'b00000000111111111111111111111111;
one[30] = 32'b00000000111111111111111111111111;
one[31] = 32'b00000000111111111111111111111111;

// TWO

two[0] = 32'b00000000000000001111000000000000;
two[1] = 32'b00000000000001100000111100000000;
two[2] = 32'b00000000001100000000011111000000;
two[3] = 32'b00000001100000000000001111110000;
two[4] = 32'b00001100000000000000001111111000;
two[5] = 32'b00011000000000000000011111111100;
two[6] = 32'b00110000000000000000111111111110;
two[7] = 32'b01100000000000000001111111111111;
two[8] = 32'b11000000000000000011111111111111;
two[9] = 32'b11000000000000000111111111111110;
two[10] = 32'b01100000000000001111111111111100;
two[11] = 32'b00011000000000011111111111111000;
two[12] = 32'b00000000000000111111111111110000;
two[13] = 32'b00000000000001111111111111100000;
two[14] = 32'b00000000000011111111111111000000;
```

```
two[15] = 32'b00000000000111111111111110000000;
two[16] = 32'b00000000001111111111111100000000;
two[17] = 32'b00000000011111111111111000000000;
two[18] = 32'b00000000111111111111110000000000;
two[19] = 32'b00000001111111111111100000000000;
two[20] = 32'b00000011111111111111000000000000;
two[21] = 32'b00000111111111111110000000000000;
two[22] = 32'b00001111111111111100000000000000;
two[23] = 32'b00011111111111111000000000000000;
two[24] = 32'b00111111111111110000000000000000;
two[25] = 32'b01111111111111100000000000000000;
two[26] = 32'b11111111111111000000000000000000;
two[27] = 32'b11111111111110000000000000000000;
two[28] = 32'b11111111111100000000000000000000;
two[29] = 32'b11111111110000000000000000000000;
two[30] = 32'b11111111111111111111111111111111;
two[31] = 32'b11111111111111111111111111111111;

//THREE
three[0] = 32'b00000000001110000000000000000000;
three[1] = 32'b00000011000000001100000000000000;
three[2] = 32'b00110000000000000111111000000000;
three[3] = 32'b11000000000000000111111111100000;
three[4] = 32'b00000000000000000111111111111000;
three[5] = 32'b00000000000000000111111111111100;
three[6] = 32'b00000000000000000111111111111110;
three[7] = 32'b00000000000000000111111111111111;
three[8] = 32'b00000000000000000111111111111111;
three[9] = 32'b00000000000000000111111111111111;
```

```
three[10] = 32'b00000000000000000111111111111111;
three[11] = 32'b00000000000000000111111111111110;
three[12] = 32'b00000000000000000111111111111100;
three[13] = 32'b00000000000000000111111111111000;
three[14] = 32'b00000000000000000111111111000000;
three[15] = 32'b00000000000000000111110000000000;
three[16] = 32'b00000000000000001100000000000000;
three[17] = 32'b00000000000000000111110000000000;
three[18] = 32'b00000000000000000111111111000000;
three[19] = 32'b00000000000000000111111111111000;
three[20] = 32'b00000000000000000111111111111100;
three[21] = 32'b00000000000000000111111111111110;
three[22] = 32'b00000000000000000111111111111111;
three[23] = 32'b00000000000000000111111111111111;
three[24] = 32'b00000000000000000111111111111111;
three[25] = 32'b00000000000000000111111111111111;
three[26] = 32'b00000000000000000111111111111110;
three[27] = 32'b11000000000000000111111111111100;
three[28] = 32'b01100000000000000111111111110000;
three[29] = 32'b00011000000000000111111100000000;
three[30] = 32'b00000001100000000110000000000000;
three[31] = 32'b00000000000111100000000000000000;

//FOUR

four[0] = 32'b00000000000000000000000000110000;
four[1] = 32'b00000000000000000000000001110000;
four[2] = 32'b00000000000000000000000011110000;
four[3] = 32'b00000000000000000000000111110000;
```

```
four[4] = 32'b00000000000000000000001111110000;
four[5] = 32'b00000000000000000000011111110000;
four[6] = 32'b00000000000000000000111111110000;
four[7] = 32'b00000000000000000001111111110000;
four[8] = 32'b00000000000000000011111111110000;
four[9] = 32'b00000000000000000111111111110000;
four[10] = 32'b00000000000000001111111111110000;
four[11] = 32'b00000000000000011111111111110000;
four[12] = 32'b00000000000000111111111111110000;
four[13] = 32'b00000000000001111111111111110000;
four[14] = 32'b00000000000011111111111111110000;
four[15] = 32'b00000000001111111111111111110000;
four[16] = 32'b00000000011111111111111111110000;
four[17] = 32'b00000000110111111111111111110000;
four[18] = 32'b00000001100111111111111111110000;
four[19] = 32'b00000011000111111111111111110000;
four[20] = 32'b00000110000111111111111111110000;
four[21] = 32'b00001100000111111111111111110000;
four[22] = 32'b00011000000111111111111111110000;
four[23] = 32'b00110000000111111111111111110000;
four[24] = 32'b01100000000111111111111111110000;
four[25] = 32'b11111111111111111111111111111111;
four[26] = 32'b11111111111111111111111111111111;
four[27] = 32'b00000000000111111111111111110000;
four[28] = 32'b00000000000111111111111111110000;
four[29] = 32'b00000000000111111111111111110000;
four[30] = 32'b00000000000111111111111111110000;
four[31] = 32'b00000000000111111111111111110000;
```

```
//FIVE

five[0] = 32'b00000001111111111111111111111111;
five[1] = 32'b00000011111111111111111111111111;
five[2] = 32'b00000110000000000000000000000000;
five[3] = 32'b00001100000000000000000000000000;
five[4] = 32'b00011000000000000000000000000000;
five[5] = 32'b00110000000000000000000000000000;
five[6] = 32'b01100000000000000000000000000000;
five[7] = 32'b11000000000000000000000000000000;
five[8] = 32'b00000011000000000000000000000000;
five[9] = 32'b00000000000011000000000000000000;
five[10] = 32'b00000000000001111110000000000000;
five[11] = 32'b00000000000001111111111100000000;
five[12] = 32'b00000000000001111111111111000000;
five[13] = 32'b00000000000001111111111111110000;
five[14] = 32'b00000000000001111111111111111000;
five[15] = 32'b00000000000001111111111111111100;
five[16] = 32'b00000000000001111111111111111110;
five[17] = 32'b00000000000001111111111111111111;
five[18] = 32'b00000000000001111111111111111111;
five[19] = 32'b00000000000001111111111111111111;
five[20] = 32'b00000000000001111111111111111111;
five[21] = 32'b00000000000001111111111111111111;
five[22] = 32'b00000000000001111111111111111110;
five[23] = 32'b00000000000001111111111111111100;
five[24] = 32'b00000000000001111111111111111000;
five[25] = 32'b00000000000001111111111111110000;
five[26] = 32'b00000000000001111111111111100000;
```

```
five[27] = 32'b00000000000001111111111110000000;
five[28] = 32'b11000000000001111111100000000000;
five[29] = 32'b00110000000001111110000000000000;
five[30] = 32'b00001100000001100000000000000000;
five[31] = 32'b00000001111100000000000000000000;

//SIX

six[0] = 32'b00000000000000000011110000000000;
six[1] = 32'b00000000000000110000000001100000;
six[2] = 32'b00000000001111110000000000001100;
six[3] = 32'b00000001111111110000000000000110;
six[4] = 32'b00000111111111110000000000000011;
six[5] = 32'b00001111111111110000000000000000;
six[6] = 32'b00011111111111110000000000000000;
six[7] = 32'b00111111111111110000000000000000;
six[8] = 32'b01111111111111110000000000000000;
six[9] = 32'b11111111111111110000000000000000;
six[10] = 32'b11111111111111110000000000000000;
six[11] = 32'b11111111111111110000000000000000;
six[12] = 32'b11111111111111110000000000000000;
six[13] = 32'b11111111111111110000000000000000;
six[14] = 32'b11111111111111110000000000000000;
six[15] = 32'b11111111111111110000000000000000;
six[16] = 32'b11111111111111110011000000000000;
six[17] = 32'b11111111111111110000000110000000;
six[18] = 32'b11111111111111110000000000110000;
six[19] = 32'b11111111111111110000000000001100;
six[20] = 32'b11111111111111110000000000000110;
```

```
six[21] = 32'b11111111111111110000000000000011;
six[22] = 32'b01111111111111110000000000000011;
six[23] = 32'b00111111111111110000000000000011;
six[24] = 32'b00011111111111110000000000000011;
six[25] = 32'b00001111111111110000000000000011;
six[26] = 32'b00000111111111110000000000000011;
six[27] = 32'b00000011111111110000000000000110;
six[28] = 32'b00000001111111110000000000001100;
six[29] = 32'b00000000001111110000000000110000;
six[30] = 32'b00000000000000011000000110000000;
six[31] = 32'b00000000000000000011110000000000;

//SEVEN

seven[0] = 32'b11111111111111111111111111111111;
seven[1] = 32'b11111111111111111111111111111111;
seven[2] = 32'b00000000000000001111111111111110;
seven[3] = 32'b00000000000000001111111111111100;
seven[4] = 32'b00000000000000011111111111111000;
seven[5] = 32'b00000000000000111111111111110000;
seven[6] = 32'b00000000000001111111111111100000;
seven[7] = 32'b00000000000011111111111111000000;
seven[8] = 32'b00000000001111111111111110000000;
seven[9] = 32'b00000000011111111111111100000000;
seven[10] = 32'b00000000111111111111111000000000;
seven[11] = 32'b00000001111111111111110000000000;
seven[12] = 32'b00000011111111111111100000000000;
seven[13] = 32'b00000111111111111111000000000000;
seven[14] = 32'b00001111111111111110000000000000;
```

```
seven[15] = 32'b00011111111111111110000000000000;
seven[16] = 32'b00111111111111111100000000000000;
seven[17] = 32'b01111111111111111000000000000000;
seven[18] = 32'b11111111111111110000000000000000;
seven[19] = 32'b11111111111111100000000000000000;
seven[20] = 32'b11111111111111000000000000000000;
seven[21] = 32'b11111111111110000000000000000000;
seven[22] = 32'b11111111111100000000000000000000;
seven[23] = 32'b11111111111000000000000000000000;
seven[24] = 32'b11111111110000000000000000000000;
seven[25] = 32'b11111111000000000000000000000000;
seven[26] = 32'b11111110000000000000000000000000;
seven[27] = 32'b11111100000000000000000000000000;
seven[28] = 32'b11111000000000000000000000000000;
seven[29] = 32'b11110000000000000000000000000000;
seven[30] = 32'b11100000000000000000000000000000;
seven[31] = 32'b11000000000000000000000000000000;

//EIGHT

eight[0] = 32'b00000000000000011000000000000000;
eight[1] = 32'b00000000000110000001100000000000;
eight[2] = 32'b00000001111111000000000110000000;
eight[3] = 32'b00001111111110000000000000110000;
eight[4] = 32'b00111111111100000000000000001100;
eight[5] = 32'b01111111111100000000000000000110;
eight[6] = 32'b11111111111110000000000000000011;
eight[7] = 32'b11111111111111000000000000000011;
eight[8] = 32'b11111111111111100000000000000011;
```

```
eight[9] = 32'b11111111111111110000000000000011;
eight[10] = 32'b11111111111111111000000000000011;
eight[11] = 32'b11111111111111111100000000000011;
eight[12] = 32'b01111111111111111110000000000110;
eight[13] = 32'b00111111111111111111000000001100;
eight[14] = 32'b00011111111111111111100001100000;
eight[15] = 32'b00001111111111111111111100000000;
eight[16] = 32'b00000111111111111111111110000000;
eight[17] = 32'b00000001111111111111111111000000;
eight[18] = 32'b00000000111111111111111111100000;
eight[19] = 32'b00000001111111111111111111110000;
eight[20] = 32'b00001100001111111111111111111000;
eight[21] = 32'b00110000000111111111111111111100;
eight[22] = 32'b01100000000011111111111111111110;
eight[23] = 32'b11000000000001111111111111111111;
eight[24] = 32'b11000000000000111111111111111111;
eight[25] = 32'b01100000000000011111111111111111;
eight[26] = 32'b00110000000000001111111111111111;
eight[27] = 32'b00011000000000000111111111111110;
eight[28] = 32'b00001100000000000011111111111100;
eight[29] = 32'b00000000110000000001111111110000;
eight[30] = 32'b00000000000110000000111100000000;
eight[31] = 32'b00000000000000001111100000000000;

//NINE

nine[0] = 32'b00000000000111000000000000000000;
nine[1] = 32'b00000001100000000110000000000000;
nine[2] = 32'b00011000000000000111111000000000;
```

```
nine[3] = 32'b01100000000000000111111111000000;
nine[4] = 32'b11000000000000000111111111110000;
nine[5] = 32'b11000000000000000111111111111000;
nine[6] = 32'b11000000000000000111111111111100;
nine[7] = 32'b11000000000000000111111111111111;
nine[8] = 32'b01100000000000000111111111111111;
nine[9] = 32'b00110000000000000111111111111111;
nine[10] = 32'b00011000000000000111111111111111;
nine[11] = 32'b00001100000000000111111111111111;
nine[12] = 32'b00000110000000000111111111111111;
nine[13] = 32'b00000000001100000111111111111111;
nine[14] = 32'b00000000000000110111111111111111;
nine[15] = 32'b00000000000000000111111111111111;
nine[16] = 32'b00000000000000000111111111111111;
nine[17] = 32'b00000000000000000111111111111111;
nine[18] = 32'b00000000000000000111111111111111;
nine[19] = 32'b00000000000000000111111111111111;
nine[20] = 32'b00000000000000000111111111111111;
nine[21] = 32'b00000000000000000111111111111111;
nine[22] = 32'b00000000000000000111111111111110;
nine[23] = 32'b00000000000000000111111111111100;
nine[24] = 32'b00000000000000000111111111111000;
nine[25] = 32'b00000000000000000111111111110000;
nine[26] = 32'b00000000000000000111111111100000;
nine[27] = 32'b11000000000000000111111111000000;
nine[28] = 32'b01100000000000000111111100000000;
nine[29] = 32'b00011000000000000111100000000000;
nine[30] = 32'b00000011000000011000000000000000;
nine[31] = 32'b00000000001110000000000000000000;
```

//MUD

```
mudmem[18] =
120'b0000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000;

mudmem[17] =
120'b0000000000000000000000000000000000000111111110000000000000000000000000000111110
00000000000000000000000000000000000000000;

mudmem[16] =
120'b0000000000000000000000000000000000111111111111111000111111000111000000111111111
11100000000000001100000000000000000000000;

mudmem[15] =
120'b0000000000000000001111111111111111111111111111111111111111111111111111111111111
11111111000011111111000000000000000000000;

mudmem[14] =
120'b0000000000000000111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111000000000000000000;

mudmem[13] =
120'b0000000000000001111111111111111111111111111111111111111111111111111111111111111
11111111111111111111110000000000000000000;

mudmem[12] =
120'b0000000000000001111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111100000000000000000;

mudmem[11] =
120'b0000000000000001111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111000000000000000000;

mudmem[10] =
120'b0000000000000011111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111000000000000000000;

mudmem[9] =
120'b0000000000001111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111100000000000000000000;

mudmem[8] =
120'b0000000000000111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111100000000000000000000;
```

```
                    mudmem[7] =
120'b000000000000011111111111111111111111100000111111111111111100000111111111100000
111111111111111111111100100000000000000000;

                    mudmem[6] =
120'b000000000000011111111111111111111111000000000110000000011000000000011000000000000
000011111100011111111111111111000000000000;

                    mudmem[5] =
120'b000000000000011111111111000000011100000000000000000000000000000000000000000000000
000000000000000000111111111100000000000000;

                    mudmem[4] =
120'b000000000000000011111111111000000000000000000000000000000000000000000000000000000
000000000000000000011111111000000000000000;

                    mudmem[3] =
120'b000000000000000011111111111000000000000000000000000000000000000000000000000000000
000000000000000000011111111000000000000000;

                    mudmem[2] =
120'b000000000000000011111111111110000000000000000000000000000000000000000000000000000
000000000000000000011111111000000000000000;

                    mudmem[1] =
120'b000000000000000000111111111100000000000000000000000000000000000000000000000000000
000000000000000000000011111100000000000000;

                    mudmem[0] =
120'b000000000000000000111111111111000000000000000000000000000000000000000000000000000
000000000000000000000111100000000000000000;


  end

  /*

   *   0 1 2 3 4 5 6 7

   *

   * 0  afa a a a ab

   * 1  f       b

   * 2  f       b

   * 3  f       b

   * 4  f       b
```

```
  * 5  f        b
  * 6 efggggggggbc
  * 7  e        c
  * 8  e        c
  * 9  e        c
  * 10 e        c
  * 11 e        c
  * 12 eddddddddc
  * 13
  * 14           h
  * 15
  *
  * 640 x 480
  *
  * Each seven-segment "pixel" is 8x8: 64 pixels across, 512 pixels for
  * 8 characters being displayed
  * 64 + 512 + 64 = 640  Start in column 64, end in column 576
  *
  * 128 pixels high: start at row 128, end at row 256
  * 128 + 128 + 224 = 480  Start in row 128
  */


          int i,j=0;
          int k = 200;


         always_comb begin


         //##########################################SKY AND GREEN GRASS BACKGROUNG
########################
```

```
        if (vcount>80)
        {VGA_R, VGA_G, VGA_B} = {8'h00, 8'h99, 8'h00};
        else
        {VGA_R, VGA_G, VGA_B} = {8'h87, 8'hce, 8'heb};


        //############################## CIRCLE EQUATION
##################################
        if ((hcount[10:1]-410)*(hcount[10:1]-410)+(vcount-450)*(vcount-450) < 100000 )  // Equation of
circle with centre = (x,y)
        {VGA_R, VGA_G, VGA_B} = {8'hca, 8'h8d, 8'h42};


        // ######################################## SHORT GRASS ############################


   for(i=0;i<=640;i=i+7)begin
                if (((vcount>=76) & (vcount<=76+7)) & ((hcount[10:1]>=i) & (hcount[10:1]<=(i+7))))begin
                        if (grassmem[vcount-76][i+7-hcount[10:1]] == 1)
                                {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
                end
        end


        //###################################### HOLES
########################################


        if ((((hcount[10:1]-290)*(hcount[10:1]-290))>>7)+(((vcount-207)*(vcount-207))>>5) <= 5'b10000
) //Hole1
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        if ((((hcount[10:1]-510)*(hcount[10:1]-510))>>7)+(((vcount-207)*(vcount-207))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
```

```
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
        if ((((hcount[10:1]-400)*(hcount[10:1]-400))>>7)+(((vcount-237)*(vcount-237))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
        if ((((hcount[10:1]-400)*(hcount[10:1]-400))>>7)+(((vcount-317)*(vcount-317))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
        if ((((hcount[10:1]-400)*(hcount[10:1]-400))>>7)+(((vcount-397)*(vcount-397))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
        if ((((hcount[10:1]-250)*(hcount[10:1]-250))>>7)+(((vcount-277)*(vcount-277))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
        if ((((hcount[10:1]-250)*(hcount[10:1]-250))>>7)+(((vcount-357)*(vcount-357))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
        if ((((hcount[10:1]-550)*(hcount[10:1]-550))>>7)+(((vcount-277)*(vcount-277))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
        if ((((hcount[10:1]-550)*(hcount[10:1]-550))>>7)+(((vcount-357)*(vcount-357))>>5) <= 5'b10000
) //Hole
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        //{VGA_R, VGA_G, VGA_B} = {8'h8b, 8'h45, 8'h13};
```

```
        //############################################ STONES
##################################################

        if (((vcount>=181) & (vcount<=(181+51))) & ((hcount[10:1]>=230) &
(hcount[10:1]<=(230+119))))begin

         if (stonemem[vcount-181][230+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end

        if (((vcount>=181) & (vcount<=(181+51))) & ((hcount[10:1]>=450) &
(hcount[10:1]<=(450+119))))begin

         if (stonemem[vcount-181][450+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end

        if (((vcount>=211) & (vcount<=(211+51))) & ((hcount[10:1]>=340) &
(hcount[10:1]<=(340+119))))begin

         if (stonemem[vcount-211][340+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end

        if (((vcount>=291) & (vcount<=(291+51))) & ((hcount[10:1]>=340) &
(hcount[10:1]<=(340+119))))begin

         if (stonemem[vcount-291][340+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end

        if (((vcount>=371) & (vcount<=(371+51))) & ((hcount[10:1]>=340) &
(hcount[10:1]<=(340+119))))begin

         if (stonemem[vcount-371][340+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end

        if (((vcount>=251) & (vcount<=(251+51))) & ((hcount[10:1]>=190) &
(hcount[10:1]<=(190+119))))begin

         if (stonemem[vcount-251][190+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};
```

```
        end

        if (((vcount>=331) & (vcount<=(331+51))) & ((hcount[10:1]>=190) &
(hcount[10:1]<=(190+119))))begin

         if (stonemem[vcount-331][190+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end

        if (((vcount>=251) & (vcount<=(251+51))) & ((hcount[10:1]>=490) &
(hcount[10:1]<=(490+119))))begin

         if (stonemem[vcount-251][490+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end

        if (((vcount>=331) & (vcount<=(331+51))) & ((hcount[10:1]>=490) &
(hcount[10:1]<=(490+119))))begin

         if (stonemem[vcount-331][490+119-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        end


        //################################### TALL
GRASS########################################


                  if (((vcount>=400) & (vcount<=(400+29))) & ((hcount[10:1]>=566) &
(hcount[10:1]<=(566+63))))

         if (grass1mem[vcount-400][63+566-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};

    if (((vcount>=400) & (vcount<=(400+39))) & ((hcount[10:1]>=566) & (hcount[10:1]<=(566+63))))

         if (grass2mem[vcount-400][63+566-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

                        if (((vcount>=440) & (vcount<=(440+29))) & ((hcount[10:1]>=500) &
(hcount[10:1]<=(500+63))))

         if (grass1mem[vcount-440][63+500-hcount[10:1]] == 1)

              {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
```

```
    if (((vcount>=440) & (vcount<=(440+39))) & ((hcount[10:1]>=500) & (hcount[10:1]<=(500+63))))
        if (grass2mem[vcount-440][63+500-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

                if (((vcount>=440) & (vcount<=(440+29))) & ((hcount[10:1]>=300) &
(hcount[10:1]<=(300+63))))
        if (grass1mem[vcount-440][63+300-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
    if (((vcount>=440) & (vcount<=(440+39))) & ((hcount[10:1]>=300) & (hcount[10:1]<=(300+63))))
        if (grass2mem[vcount-440][63+300-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};
                    if (((vcount>=440) & (vcount<=(440+29))) & ((hcount[10:1]>=250) &
(hcount[10:1]<=(250+63))))
        if (grass1mem[vcount-440][63+250-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
    if (((vcount>=440) & (vcount<=(440+39))) & ((hcount[10:1]>=250) & (hcount[10:1]<=(250+63))))
        if (grass2mem[vcount-440][63+250-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};
                    if (((vcount>=440) & (vcount<=(440+29))) & ((hcount[10:1]>=200) &
(hcount[10:1]<=(200+63))))
        if (grass1mem[vcount-440][63+200-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
    if (((vcount>=440) & (vcount<=(440+39))) & ((hcount[10:1]>=200) & (hcount[10:1]<=(200+63))))
        if (grass2mem[vcount-440][63+200-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

                    if (((vcount>=207) & (vcount<=(207+29))) & ((hcount[10:1]>=575) &
(hcount[10:1]<=(575+63))))
        if (grass1mem[vcount-207][63+575-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
```

```
    if (((vcount>=207) & (vcount<=(207+39))) & ((hcount[10:1]>=575) & (hcount[10:1]<=(575+63))))
        if (grass2mem[vcount-207][63+575-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

                if (((vcount>=185) & (vcount<=(185+29))) & ((hcount[10:1]>=200) &
(hcount[10:1]<=(200+63))))
        if (grass1mem[vcount-185][63+200-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
    if (((vcount>=185) & (vcount<=(185+39))) & ((hcount[10:1]>=200) & (hcount[10:1]<=(200+63))))
        if (grass2mem[vcount-185][63+200-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};
                if (((vcount>=207) & (vcount<=(207+29))) & ((hcount[10:1]>=150) &
(hcount[10:1]<=(150+63))))
        if (grass1mem[vcount-207][63+150-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
    if (((vcount>=207) & (vcount<=(207+39))) & ((hcount[10:1]>=150) & (hcount[10:1]<=(150+63))))
        if (grass2mem[vcount-207][63+150-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

                if (((vcount>=100) & (vcount<=(100+29))) & ((hcount[10:1]>=540) &
(hcount[10:1]<=(540+63))))
        if (grass1mem[vcount-100][63+540-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'h32, 8'hcd, 8'h32};
    if (((vcount>=100) & (vcount<=(100+39))) & ((hcount[10:1]>=540) & (hcount[10:1]<=(540+63))))
        if (grass2mem[vcount-100][63+540-hcount[10:1]] == 1)
            {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};

        //##############################  MOLE 1 ###################################
        case (hex0[3:0])
        default :if (mole1pop==0)
```

```
    begin
                mole1_x=1000;
                mole1_y=1000;
            end
            else begin
             mole1_x=temp_mole1_x;
                mole1_y=temp_mole1_y;
            end
4'h1 : begin
                mole1_x=258;
                mole1_y=141;
            end
4'h2 : begin
                mole1_x=368;
                mole1_y=171;
            end
4'h3 : begin
                mole1_x=478;
                mole1_y=141;
            end
4'h4 : begin
                mole1_x=218;
                mole1_y=217;
            end
4'h5 : begin
                mole1_x=368;
                mole1_y=251;
            end
4'h6 : begin
```

```
                mole1_x=518;
                mole1_y=217;
            end
        4'h7 : begin
                mole1_x=218;
                mole1_y=291;
            end
        4'h8 : begin
                mole1_x=368;
                mole1_y=331;
            end
        4'h9 : begin
                mole1_x=518;
                mole1_y=291;
            end
        endcase

        if (((vcount>=mole1_y+59-mole1pop) & (vcount<=(mole1_y+75))) & ((hcount[10:1]>=mole1_x) & (hcount[10:1]<=(mole1_x+63))))begin
            if (shape2mem[vcount-mole1_y-59+mole1pop][hcount[10:1]-mole1_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'h66, 8'h33, 8'h00};
            if (shape3mem[vcount-mole1_y-59+mole1pop][hcount[10:1]-mole1_x] == 0)
                {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
            if (shape4mem[vcount-mole1_y-59+mole1pop][hcount[10:1]-mole1_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
            if (shape5mem[vcount-mole1_y-59+mole1pop][hcount[10:1]-mole1_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'h0, 8'hff};
            if (shape6mem[vcount-mole1_y-59+mole1pop][hcount[10:1]-mole1_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'h00, 8'h0, 8'h0};
```

```
        if (shape7mem[vcount-mole1_y-59+mole1pop][hcount[10:1]-mole1_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
end


//################################  MOLE 2 ###################################
case (hex0[7:4])
 default : if (mole2pop==0)
     begin
                        mole2_x=1000;
                        mole2_y=1000;
                end
                else begin
                 mole2_x=temp_mole2_x;
                        mole2_y=temp_mole2_y;
                end
 4'h1 : begin
                        mole2_x=258;
                        mole2_y=141;
                end
4'h2 : begin
                        mole2_x=368;
                        mole2_y=171;
                end
4'h3 : begin
                        mole2_x=478;
                        mole2_y=141;
                end
4'h4 : begin
```

```
                        mole2_x=218;
                        mole2_y=217;
                    end
            4'h5 : begin
                        mole2_x=368;
                        mole2_y=251;
                    end
            4'h6 : begin
                        mole2_x=518;
                        mole2_y=217;
                    end
            4'h7 : begin
                        mole2_x=218;
                        mole2_y=291;
                    end
            4'h8 : begin
                        mole2_x=368;
                        mole2_y=331;
                    end
            4'h9 : begin
                        mole2_x=518;
                        mole2_y=291;
                    end
            endcase


        if (((vcount>=mole2_y+59-mole2pop) & (vcount<=(mole2_y+75))) & ((hcount[10:1]>=mole2_x)
& (hcount[10:1]<=(mole2_x+63))))begin
                if (shape2mem[vcount-mole2_y-59+mole2pop][hcount[10:1]-mole2_x] == 1)
```

```
                {VGA_R, VGA_G, VGA_B} = {8'h66, 8'h33, 8'h00};
        if (shape3mem[vcount-mole2_y-59+mole2pop][hcount[10:1]-mole2_x] == 0)
                {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
        if (shape4mem[vcount-mole2_y-59+mole2pop][hcount[10:1]-mole2_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
        if (shape5mem[vcount-mole2_y-59+mole2pop][hcount[10:1]-mole2_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'h0, 8'hff};
        if (shape6mem[vcount-mole2_y-59+mole2pop][hcount[10:1]-mole2_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'h00, 8'h0, 8'h0};
        if (shape7mem[vcount-mole2_y-59+mole2pop][hcount[10:1]-mole2_x] == 1)
                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
end

//################################  MOLE 3 ###################################
case (hex1[3:0])
 default : if (mole3pop==0)
      begin
                        mole3_x=1000;
                        mole3_y=1000;
                end
                else begin
                 mole3_x=temp_mole3_x;
                        mole3_y=temp_mole3_y;
                end
 4'h1 : begin
                        mole3_x=258;
                        mole3_y=141;
                end
4'h2 : begin
```

```
                mole3_x=368;
                mole3_y=171;
            end
4'h3 : begin
                mole3_x=478;
                mole3_y=141;
            end
4'h4 : begin
                mole3_x=218;
                mole3_y=217;
            end
4'h5 : begin
                mole3_x=368;
                mole3_y=251;
            end
4'h6 : begin
                mole3_x=518;
                mole3_y=217;
            end
4'h7 : begin
                mole3_x=218;
                mole3_y=291;
            end
4'h8 : begin
                mole3_x=368;
                mole3_y=331;
            end
4'h9 : begin
                mole3_x=518;
```

```
                    mole3_y=291;
                end
        endcase

        if (((vcount>=mole3_y+59-mole3pop) & (vcount<=(mole3_y+75))) & ((hcount[10:1]>=mole3_x)
& (hcount[10:1]<=(mole3_x+63))))begin
                if (shape2mem[vcount-mole3_y-59+mole3pop][hcount[10:1]-mole3_x] == 1)
                        {VGA_R, VGA_G, VGA_B} = {8'h66, 8'h33, 8'h00};
                if (shape3mem[vcount-mole3_y-59+mole3pop][hcount[10:1]-mole3_x] == 0)
                        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
                if (shape4mem[vcount-mole3_y-59+mole3pop][hcount[10:1]-mole3_x] == 1)
                        {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
                if (shape5mem[vcount-mole3_y-59+mole3pop][hcount[10:1]-mole3_x] == 1)
                        {VGA_R, VGA_G, VGA_B} = {8'hff, 8'h0, 8'hff};
                if (shape6mem[vcount-mole3_y-59+mole3pop][hcount[10:1]-mole3_x] == 1)
                        {VGA_R, VGA_G, VGA_B} = {8'h00, 8'h0, 8'h0};
                if (shape7mem[vcount-mole3_y-59+mole3pop][hcount[10:1]-mole3_x] == 1)
                        {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
        end


        //################################## LEVEL ######################################


         if (((vcount>=5) & (vcount<=83)) & ((hcount[10:1]>=10) & (hcount[10:1]<=169)))
       if (scoreboard1mem[vcount-5][169-hcount[10:1]] == 1)
         {VGA_R, VGA_G, VGA_B} = {8'hb6, 8'h9b, 8'h4c};
           if (((vcount>=5) & (vcount<=79)) & ((hcount[10:1]>=10) & (hcount[10:1]<=169)))
       if (scoreboardmem[vcount-5][169-hcount[10:1]] == 1)
```

```
        {VGA_R, VGA_G, VGA_B} = {8'h82, 8'h52, 8'h01};
        if (((vcount>=10) & (vcount<=25)) & ((hcount[10:1]>=40) & (hcount[10:1]<=135)))
    if (levelmem[vcount-10][135-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h33, 8'h99, 8'hff};


        //################################## SCORE ######################################
        if (((vcount>=135) & (vcount<=213)) & ((hcount[10:1]>=10) & (hcount[10:1]<=169)))
    if (scoreboard1mem[vcount-135][169-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'hb6, 8'h9b, 8'h4c};
          if (((vcount>=135) & (vcount<=209)) & ((hcount[10:1]>=10) & (hcount[10:1]<=169)))
    if (scoreboardmem[vcount-135][169-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h82, 8'h52, 8'h01};
if (((vcount>=142) & (vcount<=157)) & ((hcount[10:1]>=17) & (hcount[10:1]<=134)))
    if (scorememm[vcount-142][134-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h33, 8'h99, 8'hff};

        //################################## TIME ######################################
        if (((vcount>=275) & (vcount<=353)) & ((hcount[10:1]>=10) & (hcount[10:1]<=169)))
    if (scoreboard1mem[vcount-275][169-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'hb6, 8'h9b, 8'h4c};
          if (((vcount>=275) & (vcount<=349)) & ((hcount[10:1]>=10) & (hcount[10:1]<=169)))
    if (scoreboardmem[vcount-275][169-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h82, 8'h52, 8'h01};
if (((vcount>=280) & (vcount<=295)) & ((hcount[10:1]>=25) & (hcount[10:1]<=126)))
    if (timememm[vcount-280][126-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h33, 8'h99, 8'hff};

        //################################## W ######################################
```

```
if (((vcount>=20) & (vcount<=60)) & ((hcount[10:1]>=180) & (hcount[10:1]<=243)))
    if (momem[vcount-20][243-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

//################################## H ##########################################
if (((vcount>=20) & (vcount<=60)) & ((hcount[10:1]>=253) & (hcount[10:1]<=316)))
    if (momem2[vcount-20][316-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

//################################## A ##########################################
if (((vcount>=20) & (vcount<=60)) & ((hcount[10:1]>=326) & (hcount[10:1]<=389)))
    if (momem3[vcount-20][389-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

//################################## C ##########################################
if (((vcount>=20) & (vcount<=60)) & ((hcount[10:1]>=399) & (hcount[10:1]<=462)))
    if (mememc[vcount-20][462-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

//################################## M ##########################################
if (((vcount>=80) & (vcount<=120)) & ((hcount[10:1]>=253) & (hcount[10:1]<=316)))
    if (mmem[vcount-80][316-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

//################################## O ##########################################
if (((vcount>=80) & (vcount<=120)) & ((hcount[10:1]>=326) & (hcount[10:1]<=389)))
    if (omem[vcount-80][389-hcount[10:1]] == 1)
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};
```

```
  //################################## L ########################################
   if (((vcount>=80) & (vcount<=120)) & ((hcount[10:1]>=399) & (hcount[10:1]<=462)))
       if (lmem[vcount-80][462-hcount[10:1]] == 1)
          {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

   //################################## E ########################################
   if (((vcount>=80) & (vcount<=120)) & ((hcount[10:1]>=472) & (hcount[10:1]<=535)))
       if (emem[vcount-80][535-hcount[10:1]] == 1)
          {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

   //################################## DASH ######################################
   if (((vcount>=36) & (vcount<=51)) & ((hcount[10:1]>=472) & (hcount[10:1]<=487)))
       if (dash[vcount-36][487-hcount[10:1]] == 1)
          {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

   //################################## A ########################################
   if (((vcount>=20) & (vcount<=60)) & ((hcount[10:1]>=497) & (hcount[10:1]<=560)))
       if (momem3[vcount-20][560-hcount[10:1]] == 1)
          {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};

   //################################## DASH ######################################
   if (((vcount>=36) & (vcount<=51)) & ((hcount[10:1]>=570) & (hcount[10:1]<=585)))
       if (dash[vcount-36][585-hcount[10:1]] == 1)
          {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'hff};



          /*//DIGITS
                    if ( d_l_1_a|d_l_1_b|d_l_1_c|d_l_1_d|d_l_1_e|d_l_1_f|d_l_1_g )       // In any
(inactive) segment?
```

```
        {VGA_R, VGA_G, VGA_B} = {8'h20, 8'h20, 8'h20}; // Dark Gray   */


      //MUD TO COVER THE POPPING


//MUD TO HOLE 1
if (((vcount>=201) & (vcount<=(219))) & ((hcount[10:1]>=230) & (hcount[10:1]<=(350))))
                        if (mudmem[vcount-201][350-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};
if (((vcount>=205) & (vcount<=(223))) & ((hcount[10:1]>=230) & (hcount[10:1]<=(350))))
                        if (mudmem[vcount-205][350-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};


//MUD TO HOLE 2
if (((vcount>=231) & (vcount<=(249))) & ((hcount[10:1]>=340) & (hcount[10:1]<=(460))))
                        if (mudmem[vcount-231][460-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};
if (((vcount>=235) & (vcount<=(253))) & ((hcount[10:1]>=340) & (hcount[10:1]<=(460))))
                        if (mudmem[vcount-235][460-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};


//MUD TO HOLE 3
if (((vcount>=201) & (vcount<=(219))) & ((hcount[10:1]>=450) & (hcount[10:1]<=(570))))
                        if (mudmem[vcount-201][hcount[10:1]-570]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};
if (((vcount>=205) & (vcount<=(223))) & ((hcount[10:1]>=450) & (hcount[10:1]<=(570))))
                        if (mudmem[vcount-205][570-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};


//MUD TO HOLE 4
if (((vcount>=271) & (vcount<=(289))) & ((hcount[10:1]>=190) & (hcount[10:1]<=(310))))
```

```
                        if (mudmem[vcount-271][310-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};

if (((vcount>=275) & (vcount<=(293))) & ((hcount[10:1]>=190) & (hcount[10:1]<=(310))))

                        if (mudmem[vcount-275][310-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};

if (((vcount>=217+73) & (vcount<=217+75)) & ((hcount[10:1]>=218) & (hcount[10:1]<=218+63)))

                        {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};


//MUD TO HOLE 5

if (((vcount>=311) & (vcount<=(329))) & ((hcount[10:1]>=340) & (hcount[10:1]<=(460))))

                        if (mudmem[vcount-311][460-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};

if (((vcount>=315) & (vcount<=(333))) & ((hcount[10:1]>=340) & (hcount[10:1]<=(460))))

                        if (mudmem[vcount-315][460-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};


//MUD TO HOLE 6

if (((vcount>=271) & (vcount<=(289))) & ((hcount[10:1]>=490) & (hcount[10:1]<=(610))))

                        if (mudmem[vcount-271][610-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};

if (((vcount>=275) & (vcount<=(293))) & ((hcount[10:1]>=490) & (hcount[10:1]<=(610))))

                        if (mudmem[vcount-275][610-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};

if (((vcount>=217+73) & (vcount<=217+75)) & ((hcount[10:1]>=518) & (hcount[10:1]<=518+63)))

                        {VGA_R, VGA_G, VGA_B} = {8'ha0, 8'h52, 8'h2d};


//MUD TO HOLE 7

if (((vcount>=351) & (vcount<=(369))) & ((hcount[10:1]>=190) & (hcount[10:1]<=(310))))

                        if (mudmem[vcount-351][310-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};

if (((vcount>=355) & (vcount<=(373))) & ((hcount[10:1]>=190) & (hcount[10:1]<=(310))))
```

```
                        if (mudmem[vcount-355][310-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};


//MUD TO HOLE 8
if (((vcount>=391) & (vcount<=(409))) & ((hcount[10:1]>=340) & (hcount[10:1]<=(460))))
                        if (mudmem[vcount-391][460-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};
if (((vcount>=395) & (vcount<=(413))) & ((hcount[10:1]>=340) & (hcount[10:1]<=(460))))
                        if (mudmem[vcount-395][460-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};



//MUD TO HOLE 9
if (((vcount>=351) & (vcount<=(369))) & ((hcount[10:1]>=490) & (hcount[10:1]<=(610))))
                        if (mudmem[vcount-351][610-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};
if (((vcount>=355) & (vcount<=(373))) & ((hcount[10:1]>=490) & (hcount[10:1]<=(610))))
                        if (mudmem[vcount-355][610-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} =
{8'ha0, 8'h52, 8'h2d};



        //################################### HAMMER
########################################


        case (hex1[7:4])
         default : begin
                                hammer_x=119;
                                hammer_y=69;
                         end
        4'h1 : begin
                                hammer_x=290;
```

```
                hammer_y=207;
            end
        4'h2 : begin
                hammer_x=400;
                hammer_y=237;
            end
        4'h3 : begin
                hammer_x=510;
                hammer_y=207;
            end
        4'h4 : begin
                hammer_x=250;
                hammer_y=277;
            end
        4'h5 : begin
                hammer_x=400;
                hammer_y=317;
            end
        4'h6 : begin
                hammer_x=550;
                hammer_y=277;
            end
        4'h7 : begin
                hammer_x=250;
                hammer_y=357;
            end
        4'h8 : begin
                hammer_x=400;
                hammer_y=397;
```

```
                        end
        4'h9 : begin
                                hammer_x=550;
                                hammer_y=357;
                        end
        endcase

        if ((hex1[7:4]==0)&((vcount>=hammer_y) & (vcount<=(hammer_y+95))) &
((hcount[10:1]>=hammer_x) & (hcount[10:1]<=(hammer_x+95)))) begin
                        if (hammermem[vcount-hammer_y][hammer_x+95-hcount[10:1]] == 1)
                                {VGA_R, VGA_G, VGA_B} = {8'h99, 8'h4c, 8'h00};
                        if (hammer2mem[vcount-hammer_y][hammer_x+95-hcount[10:1]] == 1)
      {VGA_R, VGA_G, VGA_B} = {8'hcc, 8'h66, 8'h00};
        end


        if ((hex1[7:4]!=0)&((vcount>=hammer_y-117) & (vcount<=(hammer_y-22))) &
((hcount[10:1]>=hammer_x-52) & (hcount[10:1]<=(hammer_x+43)))) begin
                        if (hammermem[hcount[10:1]-hammer_x+52][vcount-hammer_y+117] == 1)
                                {VGA_R, VGA_G, VGA_B} = {8'h99, 8'h4c, 8'h00};
                        if (hammer2mem[hcount[10:1]-hammer_x+52][vcount-hammer_y+117] == 1)
      {VGA_R, VGA_G, VGA_B} = {8'hcc, 8'h66, 8'h00};
                        if ((hex1[7:4]==hex1[3:0])|(hex1[7:4]==hex0[7:4])|(hex1[7:4]==hex0[3:0]))
                          if ((vcount>=(hammer_y-53)) & (vcount<=(hammer_y-22)))
                    if    (hammer3mem[vcount-hammer_y+53][hammer_x+43-hcount[10:1]] == 1)
                                {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
        end

        //################################### SCORE DECODER
################################
```

```
        score2=hex3/100;
        score1=(hex3-score2*100)/10;
        score0=(hex3-score2*100-score1*10);

        if (((vcount>=170) & (vcount<=201)) & ((hcount[10:1]>=107) & (hcount[10:1]<=138)))
        case(score0)
                //default:;
                default:if (zero[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff,
8'hff, 8'h00};
                1:if (one[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                2:if (two[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                3:if (three[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                4:if (four[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                5:if (five[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                6:if (six[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                7:if (seven[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                8:if (eight[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                9:if (nine[vcount-170][138-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
        endcase

        if (((vcount>=170) & (vcount<=201)) & ((hcount[10:1]>=66) & (hcount[10:1]<=97)))
        case(score1)
```

```
                //default:;
                default:if (zero[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff,
8'hff, 8'h00};
                1:if (one[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                2:if (two[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                3:if (three[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                4:if (four[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                5:if (five[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                6:if (six[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                7:if (seven[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                8:if (eight[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                9:if (nine[vcount-170][97-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
        endcase

        if (((vcount>=170) & (vcount<=201)) & ((hcount[10:1]>=25) & (hcount[10:1]<=56)))
        case(score2)
                //default:;
                default:if (zero[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff,
8'hff, 8'h00};
                1:if (one[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                2:if (two[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                3:if (three[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
```

```
                4:if (four[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                5:if (five[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                6:if (six[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                7:if (seven[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                8:if (eight[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                9:if (nine[vcount-170][56-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
        endcase


        // ################################### TIME DECODER
################################

        time1=(hex4/10);

        time0=(hex4-time1*10);


        if (((vcount>=310) & (vcount<=341)) & ((hcount[10:1]>=81) & (hcount[10:1]<=112)))

        case(time0)

                //default:;

                default:if (zero[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff,
8'hff, 8'h00};

                1:if (one[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                2:if (two[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                3:if (three[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};

                4:if (four[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
```

```
                5:if (five[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                6:if (six[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                7:if (seven[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                8:if (eight[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                9:if (nine[vcount-310][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
        endcase

        if (((vcount>=310) & (vcount<=341)) & ((hcount[10:1]>=40) & (hcount[10:1]<=71)))
        case(time1)
                //default:;
                default:if (zero[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff,
8'hff, 8'h00};
                1:if (one[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                2:if (two[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
    3:if (three[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
    4:if (four[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
    5:if (five[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
    6:if (six[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
    7:if (seven[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
    8:if (eight[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
    9:if (nine[vcount-310][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
        endcase
                // ################################### LEVEL DECODER
##################################
        level1=(hex5/10);
```

```
        level0=(hex5-level1*10);


        if (((vcount>=40) & (vcount<=71)) & ((hcount[10:1]>=81) & (hcount[10:1]<=112)))
        case(level0)
                //default:;
                default:if (zero[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff,
8'hff, 8'h00};
                1:if (one[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                2:if (two[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                3:if (three[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                4:if (four[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                5:if (five[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                6:if (six[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                7:if (seven[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                8:if (eight[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                9:if (nine[vcount-40][112-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
        endcase


        if (((vcount>=40) & (vcount<=71)) & ((hcount[10:1]>=40) & (hcount[10:1]<=71)))
        case(level1)
                //default:;
                default:if (zero[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
```

```
                1:if (one[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                2:if (two[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                3:if (three[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                4:if (four[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                5:if (five[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                6:if (six[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'h00};
                7:if (seven[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                8:if (eight[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
                9:if (nine[vcount-40][71-hcount[10:1]]==1) {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff,
8'h00};
        endcase




end

  /* logic                         inChar; // In any character

  assign inChar = (vcount[9:7] == 3'd1) &
                  (hcount[10:7] != 4'd0 & hcount[10:7] != 4'd9);

  logic [2:0]                   charx; // Coordinate within the 8x16 char
  logic [3:0]                   chary;
```

```
assign charx = hcount[6:4];
assign chary = vcount[6:3];

logic horizBar, leftCol, rightCol, topCol, botCol; // Parts of the disp.

assign horizBar = !(charx[2:1] == 2'b11);  // When in any horizontal bar
assign leftCol  = (charx == 3'd0);        // When in left column
assign rightCol = (charx == 3'd5);        // When in right column
assign topCol   = !chary[3] & !(chary[2:0] == 3'd7); // Top columns
assign botCol   = (chary >= 4'd6) & (chary <= 4'd12); // Bottom columns

logic [7:0] segment; // True when in each segment
assign segment[0] = horizBar & (chary == 4'd 0);
assign segment[1] = rightCol & topCol;
assign segment[2] = rightCol & botCol;
assign segment[3] = horizBar & (chary == 4'd 12);
assign segment[4] = leftCol & botCol;
assign segment[5] = leftCol & topCol;
assign segment[6] = horizBar & (chary == 4'd 6);
assign segment[7] = (charx == 3'd6) & (chary == 4'd14);

logic [2:0] column; // Being displayed
assign column = hcount[9:7];

logic [7:0] curSegs;
assign curSegs = column == 3'd1 ? hex0 :
                 column == 3'd2 ? hex1 :
                 column == 3'd3 ? hex2 :
                 column == 3'd4 ? hex3 :
```

```
                    column == 3'd5 ? hex4 :
                    column == 3'd6 ? hex5 :
                    column == 3'd7 ? hex6 :
                    hex7; //


  always_comb begin
    {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0}; // Black
    if (inChar)
          if ( |(curSegs & segment) ) // In any active segment?
            {VGA_R, VGA_G, VGA_B} = {8'hff, 8'h00, 8'h00}; // Red
          else if ( |segment )       // In any (inactive) segment?
            {VGA_R, VGA_G, VGA_B} = {8'h20, 8'h20, 8'h20}; // Dark Gray
  end
          */


endmodule // VGA_LED_Emulator
```

hello.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "vga_led.h"
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include "usbkeyboard.h"
#include <time.h>
```

```
int vga_led_fd;
int timer=0;
int level=1;
////LIBUSB DECLARATIONS
struct usb_keyboard_packet packet;
int transferred;
char keystate[12];

///TIME DECLARATIONS
unsigned int x_hours=0;
unsigned int x_minutes=0;
unsigned int x_seconds=0;
unsigned int x_milliseconds=0;
unsigned int totaltime=0,count_down_time_in_secs=60,time_left=0;


struct libusb_device_handle *keyboard;
uint8_t endpoint_address;
//unsigned char endpoint_address;

/* Read and print the segment values */
void print_segment_info() {
  vga_led_arg_t vla;
  int i;

  for (i = 0 ; i < VGA_LED_DIGITS ; i++) {
    vla.digit = i;
    if (ioctl(vga_led_fd, VGA_LED_READ_DIGIT, &vla)) {
      perror("ioctl(VGA_LED_READ_DIGIT) failed");
      return;
    }
    printf("%03x ", vla.segments);
  }
  printf("\n");
}

/* Write the contents of the array to the display */
void write_segments(unsigned int segs[2])
{
  vga_led_arg_t vla;
  int i;
  for (i = 0 ; i < VGA_LED_DIGITS ; i++) {
    vla.digit = i;
    vla.segments = segs[i];
    //ioctl(vga_led_fd, VGA_LED_WRITE_DIGIT, &vla);
    if (ioctl(vga_led_fd, VGA_LED_WRITE_DIGIT, &vla)) {
      //perror("ioctl(VGA_LED_WRITE_DIGIT) failed");
      return;
    }
  }
```

```
}



///Time Thread
pthread_t time_thread;
void *time_thread_f(void *ignored)
{ clock_t x_startTime,x_countTime;


   x_startTime=clock();  // start clock
   time_left=count_down_time_in_secs-x_seconds;   // update timer

   while (time_left>=0)
    {
      x_countTime=clock(); // update timer difference
      x_milliseconds=x_countTime-x_startTime;
      //x_seconds=(x_milliseconds/(CLOCKS_PER_SEC))-(x_minutes*60);
      //x_minutes=(x_milliseconds/(CLOCKS_PER_SEC))/60;
      x_seconds=((x_milliseconds)/2026000)-(x_minutes*60);
      x_minutes=(((x_milliseconds)/2026000)/60);
      x_hours=x_minutes/60;



      time_left=count_down_time_in_secs-x_seconds; // subtract to get
difference
                  if (time_left==0) {
            time_left=60;
            level++;
      }

      //printf( "\nYou have %d seconds left ( %d ) count down timer by
TopCoder",time_left,count_down_time_in_secs);
    }


    //printf( "\n\n\nTime's out\n\n\n");

  }

///Keyboard Thread

pthread_t keyboard_thread;
void *keyboard_thread_f(void *ignored)
{
   while(1){
    libusb_interrupt_transfer(keyboard, endpoint_address,
```

```
                (unsigned char *) &packet, sizeof(packet),
                &transferred, 0);
    /*if (transferred == sizeof(packet)){
      sprintf(keystate, "%02x %02x %02x", packet.modifiers,
packet.keycode[0],
          packet.keycode[1]);
      printf("%s\n", keystate);}*/
    }

  }

/////////
int main()
{
  int c0 = 1, c2 = 1; //Counters used for incrementing and
decrementing the x,y coordinates of the ball
  vga_led_arg_t vla;
  int score=0; //score for the game
  int i,flag1=0,flag2=0,flag3=0,flag_level=1;
  int mole1,mole2,mole3; //moles
  static const char filename[] = "/dev/vga_led";

  static unsigned int message[8] = { 0x00, 0x00
, 0x00, 0x00,           // Initialize position of the ball to upper
left corner just like in the driver
                      0x00, 0x00, 0x00, 0x00 };


  //char keystate[12];
//lab2

  /* Open the keyboard */
//lab2
  if ( (keyboard = openkeyboard(&endpoint_address)) == NULL ) {
//lab2
    fprintf(stderr, "Did not find a keyboard\n");
//lab2
    exit(1);
//lab2
  }
//lab2

  printf("VGA LED Userspace program started\n");

  if ( (vga_led_fd = open(filename, O_RDWR)) == -1) {
    fprintf(stderr, "could not open %s\n", filename);
    return -1;
  }

  //printf("initial state: ");
```

```
  //print_segment_info();


  /* Start the time thread */
 pthread_create(&time_thread, NULL, time_thread_f, NULL);


 /* Start the keyboard thread */
 pthread_create(&keyboard_thread, NULL, keyboard_thread_f, NULL);


for(;;) {

    message[4]=time_left;

    if (flag_level!=level){
         flag_level=level;
     usleep(1500000);
    }

    if(level==2) message[2]=1;
    message[5]=level;

    if ((x_seconds%3==0)&(flag1!=x_seconds)){
    flag1=x_seconds;
    mole1 = rand() % 10;
    if ((mole1==mole2) | (mole1==mole3)) mole1=0;
    mole2=0;
    }
    if (((x_seconds-1)%3==0)&(flag2!=x_seconds-1)){
    flag2=x_seconds-1;
    mole2 = rand() % 10;
    if ((mole2==mole1) | (mole2==mole3)) mole2=0;
    mole3=0;
    }
    if (((x_seconds-2)%3==0)&(flag3!=x_seconds-2)){
    flag3=x_seconds-2;
    mole3 = rand() % 10;
    if ((mole3==mole1) | (mole3==mole2)) mole3=0;
    mole1=0;
    }

    //printf("Mole positions are:MOLE1:%d MOLE2:%d
MOLE3:%d\n",mole1,mole2,mole3);
    message[0] = ((mole2<<4) | mole1);
    message[1] = (message[1] & 0xf0);
    message[1] = (message[1] | mole3);
```

```
      switch (packet.keycode[0]){
    default:{   //printf("The packet.keycode[0]
is:%d\n",packet.keycode[0]);
          message[1] = (message[1] & 0x0f);
          message[1] = (message[1] | mole3);
          write_segments(message);
     break;}
    case 0x1e:{
          message[1] = (message[1] & 0x0f);
          message[1] = (message[1] | 0x10);
          write_segments(message);
         usleep(150000);
                    if ((mole1==1) | (mole2==1) | (mole3==1)){
           score++;
                         message[3]=score;
           if (mole1==1) mole1=0;
              if (mole2==1) mole2=0;
           if (mole3==1) mole3=0;
          }
          break;}
    case 0x1f:{
                    message[1] = (message[1] & 0x0f);
          message[1] = (message[1] | 0x20);
          write_segments(message);
         usleep(150000);
         if ((mole1==2) | (mole2==2) | (mole3==2)){
           score++;
                    message[3]=score;
           if (mole1==2) mole1=0;
              if (mole2==2) mole2=0;
           if (mole3==2) mole3=0;
          }
          break;}
    case 0x20:{message[1] = (message[1] & 0x0f);
          message[1] = (message[1] | 0x30);
          write_segments(message);
         usleep(150000);
         if ((mole1==3) | (mole2==3) | (mole3==3)){
           score++;
                         message[3]=score;
           if (mole1==3) mole1=0;
              if (mole2==3) mole2=0;
           if (mole3==3) mole3=0;
          }
          break;}
    case 0x21:{ message[1] = (message[1] & 0x0f);
          message[1] = (message[1] | 0x40);
          write_segments(message);
         usleep(150000);
```

```
     if ((mole1==4) | (mole2==4) | (mole3==4)){
       score++;
                     message[3]=score;
       if (mole1==4) mole1=0;
          if (mole2==4) mole2=0;
       if (mole3==4) mole3=0;
      }
      break;}
case 0x22:{ message[1] = (message[1] & 0x0f);
      message[1] = (message[1] | 0x50);
      write_segments(message);
     usleep(150000);
     if ((mole1==5) | (mole2==5) | (mole3==5)){
       score++;
                     message[3]=score;
       if (mole1==5) mole1=0;
          if (mole2==5) mole2=0;
       if (mole3==5) mole3=0;
      }
      break;}
case 0x23:{ message[1] = (message[1] & 0x0f);
      message[1] = (message[1] | 0x60);
      write_segments(message);
     usleep(150000);
     if ((mole1==6) | (mole2==6) | (mole3==6)){
       score++;
                     message[3]=score;
       if (mole1==6) mole1=0;
          if (mole2==6) mole2=0;
       if (mole3==6) mole3=0;
      }
      break;}
case 0x24:{ message[1] = (message[1] & 0x0f);
      message[1] = (message[1] | 0x70);
      write_segments(message);
     usleep(150000);
     if ((mole1==7) | (mole2==7) | (mole3==7)){
       score++;
                     message[3]=score;
       if (mole1==7) mole1=0;
          if (mole2==7) mole2=0;
       if (mole3==7) mole3=0;
      }
      break;}
case 0x25:{ message[1] = (message[1] & 0x0f);
      message[1] = (message[1] | 0x80);
      write_segments(message);
     usleep(150000);
     if ((mole1==8) | (mole2==8) | (mole3==8)){
       score++;
```

```
                        message[3]=score;
            if (mole1==8) mole1=0;
               if (mole2==8) mole2=0;
            if (mole3==8) mole3=0;
           }
           break;}
    case 0x26:{ message[1] = (message[1] & 0x0f);
           message[1] = (message[1] | 0x90);
           write_segments(message);
          usleep(150000);
          if ((mole1==9) | (mole2==9) | (mole3==9)){
            score++;
                        message[3]=score;
            if (mole1==9) mole1=0;
               if (mole2==9) mole2=0;
            if (mole3==9) mole3=0;
           }
           break;}
    }


 }

///LAB3 CODE FOR THE BALL...WE KEEP IT BECAUSE WE WORKED FOR IT
    /*while (1) {
//Add c0 counter(initialized as 1) to message[0](x coordinate) and c2
counter to
    message[0] = message[0]+c0;
//message[2] (y coordinate).message[1] and message[3] represent the
MSD hex digit
    message[1] = message[0]>>8;
// of the x and y coordinates respectively so they are shifted by 8
bits
    message[2] = message[2]+c2;
    message[3] = message[2]>>8;
//Boundary conditions on the screen:
    if ((message[0] == 0x63) && (message[1] == 0x00))
//If left boundary is hit then increment x
     c0 = 1;
    else if ((message[0] == 0x21D) &&(message[1] == 0X02))
//If right boundary is hit then decrement x
    c0 = (-1);
    if ((message[2] == 0x63) && (message[3] == 0x00))
//If top boundary is hit then increment y
    c2 = (1);
    else if ((message[2] == 0x17D) && (message[3] == 0x01))
//If bottom boundary is hit then decrement y
    c2 = (-1);
    write_segments(message);
    printf("current state x=%02x %02x, y=%02x %02x \n \n \n",
```

```
message[1], message[0], message[3], message[2]);    //print
coordinates on the screen
     usleep(10000);                  //For controlling the speed of
Ball Movement
    }*/

  /* Terminate the keyboard thread */
  pthread_cancel(keyboard_thread);

  /* Terminate the time thread */
  pthread_cancel(time_thread);

  /* Wait for the keyboard thread to finish */
  pthread_join(keyboard_thread, NULL);

  /* Wait for the time thread to finish */
  pthread_join(time_thread, NULL);

  printf("VGA LED Userspace program terminating\n");

  return 0;
}
```

usbkeyboard.c

```
#include "usbkeyboard.h"


#include <stdio.h>

#include <stdlib.h>


/* References on libusb 1.0 and the USB HID/keyboard protocol

 *

 * http://libusb.org

 * http://www.dreamincode.net/forums/topic/148707-introduction-to-using-libusb-10/

 * http://www.usb.org/developers/devclass_docs/HID1_11.pdf

 * http://www.usb.org/developers/devclass_docs/Hut1_11.pdf

 */


/*
```

```
 * Find and return a USB keyboard device or NULL if not found
 * The argument con
 *
 */
struct libusb_device_handle *openkeyboard(uint8_t *endpoint_address) {
  libusb_device **devs;
  struct libusb_device_handle *keyboard = NULL;
  struct libusb_device_descriptor desc;
  ssize_t num_devs, d;
  uint8_t i, k;

  /* Start the library */
  if ( libusb_init(NULL) < 0 ) {
    fprintf(stderr, "Error: libusb_init failed\n");
    exit(1);
  }

  /* Enumerate all the attached USB devices */
  if ( (num_devs = libusb_get_device_list(NULL, &devs)) < 0 ) {

    fprintf(stderr, "Error: libusb_get_device_list failed\n");
    exit(1);
  }
printf("enumerated library\n");
printf("%d\n", num_devs);
  /* Look at each device, remembering the first HID device that speaks
     the keyboard protocol */

  for (d = 0 ; d < num_devs ; d++) {
```

```
libusb_device *dev = devs[d];
if ( libusb_get_device_descriptor(dev, &desc) < 0 ) {
  fprintf(stderr, "Error: libusb_get_device_descriptor failed\n");
  exit(1);
}

if (desc.bDeviceClass == LIBUSB_CLASS_PER_INTERFACE) {
  struct libusb_config_descriptor *config;
  libusb_get_config_descriptor(dev, 0, &config);
  for (i = 0 ; i < config->bNumInterfaces ; i++)
         for ( k = 0 ; k < config->interface[i].num_altsetting ; k++ ) {
           const struct libusb_interface_descriptor *inter =
             config->interface[i].altsetting + k ;
         printf("interface class, protocol\n");
         printf("%d %d\n", inter->bInterfaceClass, inter->bInterfaceProtocol);
           if ( inter->bInterfaceClass == LIBUSB_CLASS_HID)
               //inter->bInterfaceProtocol == USB_HID_KEYBOARD_PROTOCOL) {
             {int r;
       printf("Found the keyboard\n");
             if ((r = libusb_open(dev, &keyboard)) != 0) {
               fprintf(stderr, "Error: libusb_open failed: %d\n", r);
               exit(1);
             }
             if (libusb_kernel_driver_active(keyboard,i)){
                     printf("KERNEL ACTIVE\n");
               libusb_detach_kernel_driver(keyboard, i);}
             //libusb_detach_kernel_driver(keyboard, i);
             if ((r = libusb_claim_interface(keyboard, i)) != 0) {
               fprintf(stderr, "Error: libusb_claim_interface failed: %d\n", r);
```

```
            exit(1);
          }
        printf("Claimed the interface endpoint %d\n",  inter->endpoint[0].bEndpointAddress);
          *endpoint_address = inter->endpoint[0].bEndpointAddress;
          goto found;
        }
       }
  }
 }

found:
 libusb_free_device_list(devs, 1);

 return keyboard;
}
```

usbkeyboard.h

```
#ifndef _USBKEYBOARD_H
#define _USBKEYBOARD_H

#include <libusb-1.0/libusb.h>

#define USB_HID_KEYBOARD_PROTOCOL 1

/* Modifier bits */
#define USB_LCTRL  (1 << 0)
#define USB_LSHIFT (1 << 1)
#define USB_LALT   (1 << 2)
#define USB_LGUI   (1 << 3)
```

```
#define USB_RCTRL  (1 << 4)
#define USB_RSHIFT (1 << 5)
#define USB_RALT   (1 << 6)
#define USB_RGUI   (1 << 7)

struct usb_keyboard_packet {
  uint8_t modifiers;
  uint8_t reserved;
  uint8_t keycode[6];
};

/* Find and open a USB keyboard device.  Argument should point to
   space to store an endpoint address.  Returns NULL if no keyboard
   device was found. */
extern struct libusb_device_handle *openkeyboard(uint8_t *);

#endif
```

vga_led.c

```
/*
 * Device driver for the VGA LED Emulator
 *
 * A Platform device implemented using the misc subsystem
 *
 * Stephen A. Edwards
 * Columbia University
 *
 * References:
```

```
 * Linux source: Documentation/driver-model/platform.txt
 *               drivers/misc/arm-charlcd.c
 * http://www.linuxforu.com/tag/linux-device-drivers/
 * http://free-electrons.com/docs/
 *
 * "make" to build
 * insmod vga_led.ko
 *
 * Check code style with
 * checkpatch.pl --file --no-tree vga_led.c
 */

//########################################  GEORGIOS CHARITOS AND ADITYA BAGRI
######################################################################

#include <linux/module.h>
#include <linux/init.h>
#include <linux/errno.h>
#include <linux/version.h>
#include <linux/kernel.h>
#include <linux/platform_device.h>
#include <linux/miscdevice.h>
#include <linux/slab.h>
#include <linux/io.h>
#include <linux/of.h>
#include <linux/of_address.h>
#include <linux/fs.h>
#include <linux/uaccess.h>
#include "vga_led.h"
```

```
#define DRIVER_NAME "vga_led"

/*
 * Information about our device
 */

struct vga_led_dev {
	struct resource res; /* Resource: our registers */
	void __iomem *virtbase; /* Where registers can be accessed in memory */
	u8 segments[VGA_LED_DIGITS];
} dev;

/*
 * Write segments of a single digit
 * Assumes digit is in range and the device information has been set up
 */

static void write_digit(int digit, u8 segments)
{
	iowrite8(segments, dev.virtbase + digit);
	dev.segments[digit] = segments;
}

/*
 * Handle ioctl() calls from userspace:
 * Read or write the segments on single digits.
 * Note extensive error checking of arguments
 */
```

```
static long vga_led_ioctl(struct file *f, unsigned int cmd, unsigned long arg)
{
	vga_led_arg_t vla;

	switch (cmd) {
	case VGA_LED_WRITE_DIGIT:
		if (copy_from_user(&vla, (vga_led_arg_t *) arg,
					sizeof(vga_led_arg_t)))
			return -EACCES;
		if (vla.digit > 8)
			return -EINVAL;
		write_digit(vla.digit, vla.segments);
		break;

	case VGA_LED_READ_DIGIT:
		if (copy_from_user(&vla, (vga_led_arg_t *) arg,
					sizeof(vga_led_arg_t)))
			return -EACCES;
		if (vla.digit > 8)
			return -EINVAL;
		vla.segments = dev.segments[vla.digit];
		if (copy_to_user((vga_led_arg_t *) arg, &vla,
					sizeof(vga_led_arg_t)))
			return -EACCES;
		break;

	default:
		return -EINVAL;
```

```
	}

	return 0;
}

/* The operations our device knows how to do */

static const struct file_operations vga_led_fops = {
	.owner		= THIS_MODULE,
	.unlocked_ioctl = vga_led_ioctl,
};

/* Information about our device for the "misc" framework -- like a char dev */

static struct miscdevice vga_led_misc_device = {
	.minor		= MISC_DYNAMIC_MINOR,
	.name		= DRIVER_NAME,
	.fops		= &vga_led_fops,
};

/*
 * Initialization code: get resources (registers) and display
 * a welcome message
 */

static int __init vga_led_probe(struct platform_device *pdev)
{
	static unsigned char welcome_message[VGA_LED_DIGITS] = {    //Put ball in the upper left cornern when the driver is inserted with insmod
```

```
		0x00, 0X00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};   //welcome_message[1] and welcome
message[0] are concatenated in the hardware
                                                                //welcome_message[3] and welcome
message[2] are concatenated in the hardware
	int i, ret;

	/* Register ourselves as a misc device: creates /dev/vga_led */
	ret = misc_register(&vga_led_misc_device);

	/* Get the address of our registers from the device tree */
	ret = of_address_to_resource(pdev->dev.of_node, 0, &dev.res);
	if (ret) {
		ret = -ENOENT;
		goto out_deregister;
	}

	/* Make sure we can use these registers */
	if (request_mem_region(dev.res.start, resource_size(&dev.res),
				DRIVER_NAME) == NULL) {
		ret = -EBUSY;
		goto out_deregister;
	}

	/* Arrange access to our registers */
	dev.virtbase = of_iomap(pdev->dev.of_node, 0);
	if (dev.virtbase == NULL) {
		ret = -ENOMEM;
		goto out_release_mem_region;
	}
```

```
	/* Display a welcome message */
	for (i = 0; i < VGA_LED_DIGITS; i++)
		write_digit(i, welcome_message[i]);

	return 0;

out_release_mem_region:
	release_mem_region(dev.res.start, resource_size(&dev.res));
out_deregister:
	misc_deregister(&vga_led_misc_device);
	return ret;
}

/* Clean-up code: release resources */
static int vga_led_remove(struct platform_device *pdev)
{
	iounmap(dev.virtbase);
	release_mem_region(dev.res.start, resource_size(&dev.res));
	misc_deregister(&vga_led_misc_device);
	return 0;
}

/* Which "compatible" string(s) to search for in the Device Tree */
#ifdef CONFIG_OF
static const struct of_device_id vga_led_of_match[] = {
	{ .compatible = "altr,vga_led" },
	{},
};
```

```
MODULE_DEVICE_TABLE(of, vga_led_of_match);
#endif

/* Information for registering ourselves as a "platform" driver */
static struct platform_driver vga_led_driver = {
	.driver	= {
		.name	= DRIVER_NAME,
		.owner	= THIS_MODULE,
		.of_match_table = of_match_ptr(vga_led_of_match),
	},
	.remove		= __exit_p(vga_led_remove),
};

/* Called when the module is loaded: set things up */
static int __init vga_led_init(void)
{
	pr_info(DRIVER_NAME ": init\n");
	return platform_driver_probe(&vga_led_driver, vga_led_probe);
}

/* Called when the module is unloaded: release resources */
static void __exit vga_led_exit(void)
{
	platform_driver_unregister(&vga_led_driver);
	pr_info(DRIVER_NAME ": exit\n");
}

module_init(vga_led_init);
module_exit(vga_led_exit);
```

```
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Stephen A. Edwards, Columbia University");
MODULE_DESCRIPTION("VGA 7-segment LED Emulator");
```

vga_led.h

```
#ifndef _VGA_LED_H
#define _VGA_LED_H

#include <linux/ioctl.h>

#define VGA_LED_DIGITS 8

typedef struct {
  unsigned char digit;    /* 0, 1, .. , VGA_LED_DIGITS - 1 */
  unsigned char segments; /* LSB is segment a, MSB is decimal point */
} vga_led_arg_t;

#define VGA_LED_MAGIC 'q'

/* ioctls and their arguments */
#define VGA_LED_WRITE_DIGIT _IOW(VGA_LED_MAGIC, 1, vga_led_arg_t *)
#define VGA_LED_READ_DIGIT  _IOWR(VGA_LED_MAGIC, 2, vga_led_arg_t *)

#endif
```

Acknowledgement

We would like to take this opportunity to thank Professor Stephen Edwards for such an interesting and intellectually stimulating Embedded Systems course, we sure learned a great deal from it about hardware and software. Also, Prof. Edwards was constantly available for advice, guidance and assistance during the course of this project and we are extremely grateful for that.

We would also like to thank the three TA's for this course for all their help during the assignments as well as the project work.