Project Report: Awesome Guitar Game

CSEE4840 Embedded System Design

Columbia University, Spring 2012

Laurent Charignon(lc2817)

Avijit Singh Wasu(asw2156)

I Overview

This game is inspired by the game Guitar Hero video game series . We use Guitar

Hero controller for PlayStation 2 to serve as the "dummy guitar", it has 5 colored buttons which correspond to the notes that are displayed on the screen. When the user starts the game he/she hears the music. The screen displays a stream of notes. If the user presses the correct button at the correct instant his score is increased. To make the user more involved, the display will reflect that the note has been correctly played.

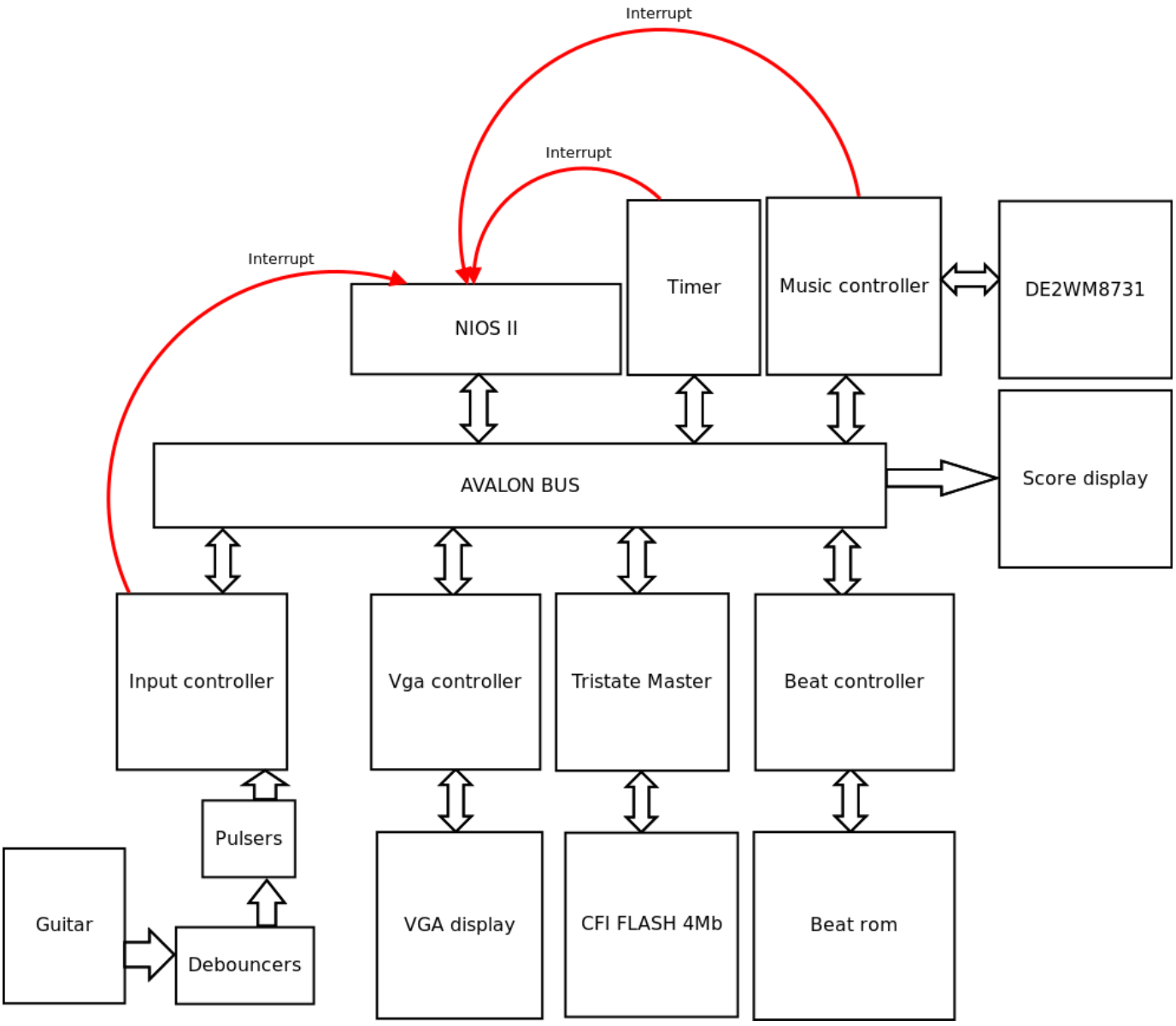
To implement the game, the project involves both hardware setup and software programming. The hardware is developed in VHDL and the software for the microcontroller is written in the C programming language.We have rebuild the circuitry of the PS2 Guitar controller to make it work with the FPGA.We connect the Guitar to a mini intermediary circuitry which is in turn connected to the GPIOs on the FPGA via a ribbon cable.

The game starts with the press of any note of the guitar.The music plays and the notes stream on the five strings on the screen.The user is required to press the corresponding color note on the Guitar as he sees on the screen.A track of the score is kept and is displayed on the upper right corner of the screen.The score varies between one to five depending on the performance of the user.The precise score displaying the exact number of correct keys pressed is displayed on the hex display on the board.

I Design and implementation

II.1 Architecture

Here is the global architecture of our project:

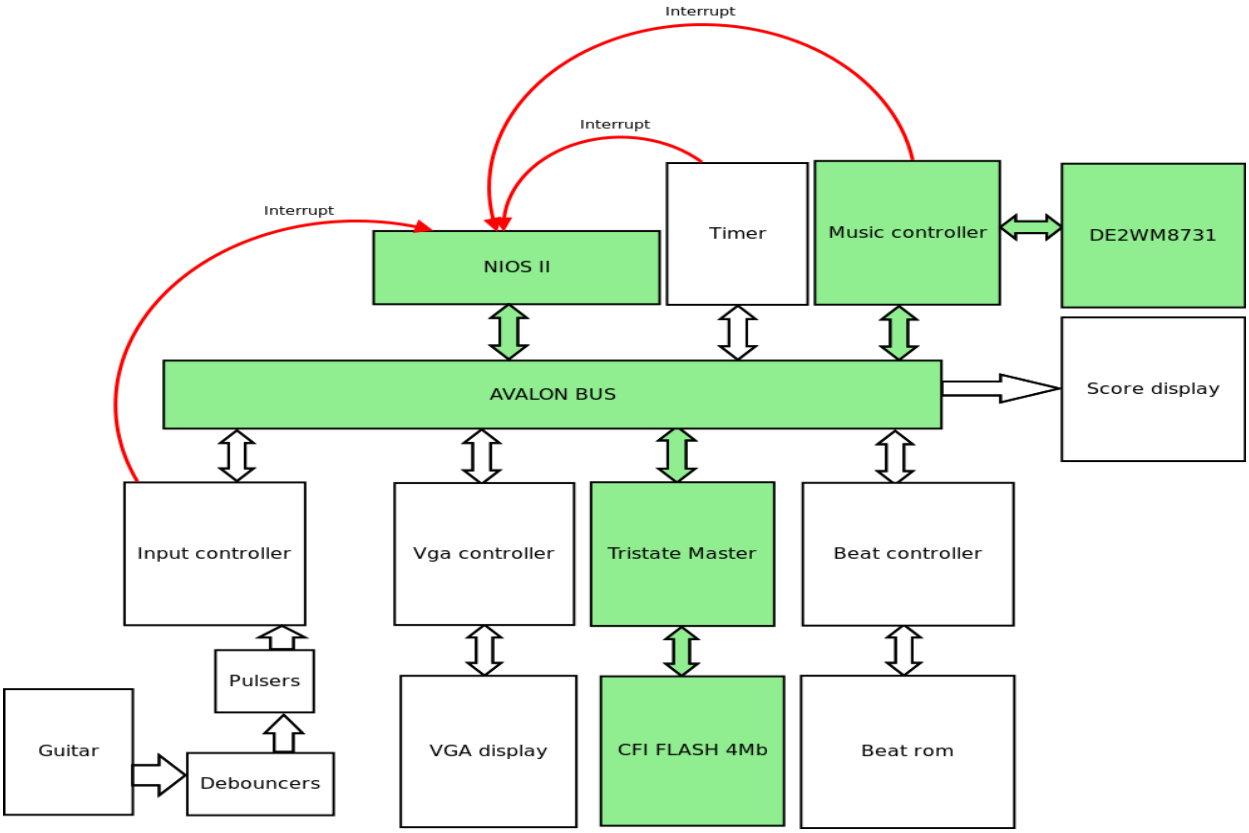


We will follow the previous diagram and explore it through three aspects of our project, introducing the components in a logical order:

- How we play the music
- How we get the user input
- How we deal with the falling beats of the song

II.2 How we play the music

The schema below highlight the components at stakes while playing a music



II.2.1 The flash memory

Introduction

We have faced the necessity to store an entire sound track on the board. We have considered several storage solutions: the SDRAM, the flash memory and the SDRAM.
We have decided to finally use the flash since :

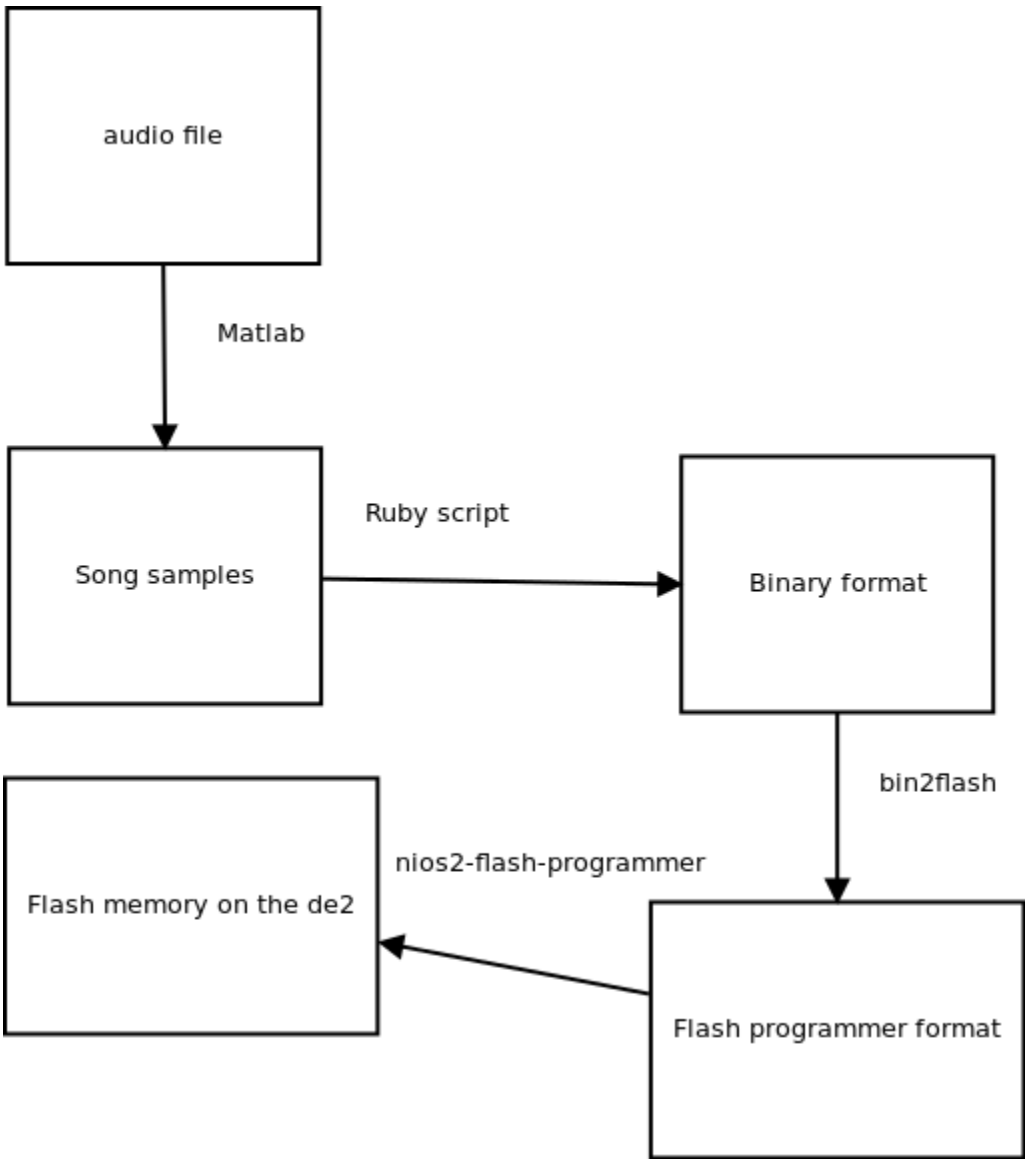
- it is very easy to transfer data to it
- the memory is not volatile (we don't have to reload the song every time we want to test a new design)
- we haven't had to write much code to make it work and relied a lot on the builtin components

Architecture

We have followed the guidelines of this guide to use the flash and the flash-programmer tool with the nios processor. So we have a tristate master component that makes the bridge between the avalon bus and the flash. We have found the right "timing parameters of the flash" on an internet website and we have forgotten to note its name. Please note: we have communicated those timing information to other groups willing to use flash as they have been quite difficult to find.

Usage and conversion

To transfer the data to the flash we have used two command line tools: `bin2flash` and `nios2-flash-programmer` and a ruby script to do the conversion to binary format.



From Matlab we have a song in the following format:

`Rhapsd.txt`

```
100 114 120 119 122 131 134 110 96 120 123 119 127 117 126 122 125 120 127
```

```
134 139 147 128 113 12 ...
```

All these numbers are on 8 bits and represents the samples of the song played at 16khz.
We use the following Ruby script to transfer it in binary format:

toBinForFlash.rb

```
inf, out = File.open("Rhapsd.txt"), File.open("song.bin","wb")
 # Integer#pack("C") gets the binary representation (unsigned 8 bit int)
 # of the number
 inf.read.split(" ").each {|sample| out.write([Integer(sample)].pack('C'))}
inf.close ; out.close
```

And then we call bin2flash to convert the binary song into the a file that the flash programmer can send over to the flash:

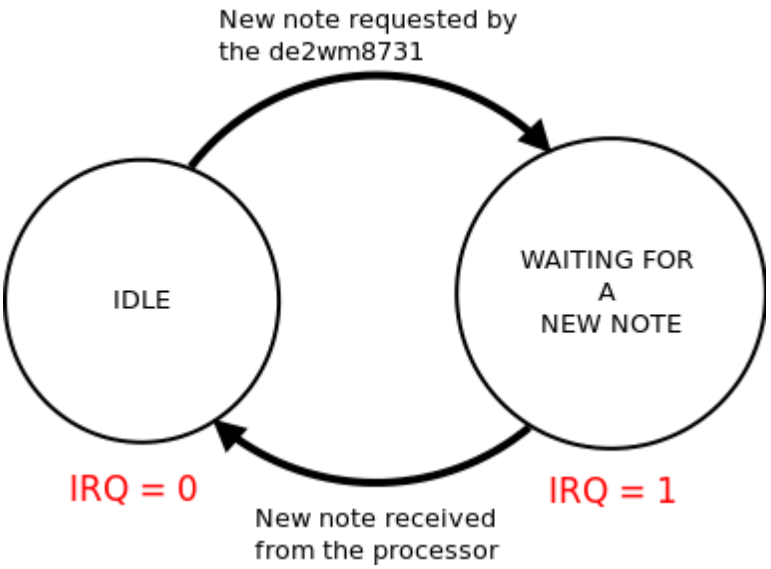
```
bin2flash --input=song.bin --output=song.flash --location=0
```

Here we put location=0 since the location is relative to the start of the flash memory. The last step is to transfer the music to the flash memory. After having transferred a design with a nios and a flash controller properly configured, we can put the song in the flash memory. Assuming that the flash memory starts at the address 0x400000 we use
`nios2-flash-programmer -b 0x400000 --program song.flash`

Once the music is put in the flash, we can retrieve it sample by sample and play it by using the music controller. That is the subject of the following part.

II.2.2 The music controller

To play sound tracks on the board, we have decided to reuse the components used in the lab 3 wrapping them in an avalon module to enable the processor to send new notes when required. We have the following final state machine for the avalon component:



The de2wm8731 has been changed a little to adopt new sampling rates. We have tried 8khz and 16khz (which are both acceptable in terms of memory dimensioning) but we are finally using 16khz, 8 bits per sample.

II.2.3 Synchronizing everything

On the software we make the flash memory and the music controller work together and play the music.
The design of our project code is explained below:

The software part (the program that runs on the NIOS II) is made of three parts:

- A set of callbacks for interruptions
- An initialization phase where we mostly trigger the interruption and initialize values
- An infinite loop that executes various operations

To play the music we have:

- Registered a callback for the interruptions of the music controller
- Written a callback that sends a new note to the music controller when it requested it
- Put some code in the infinite loop to constantly have the next note ready when requested by the music controller

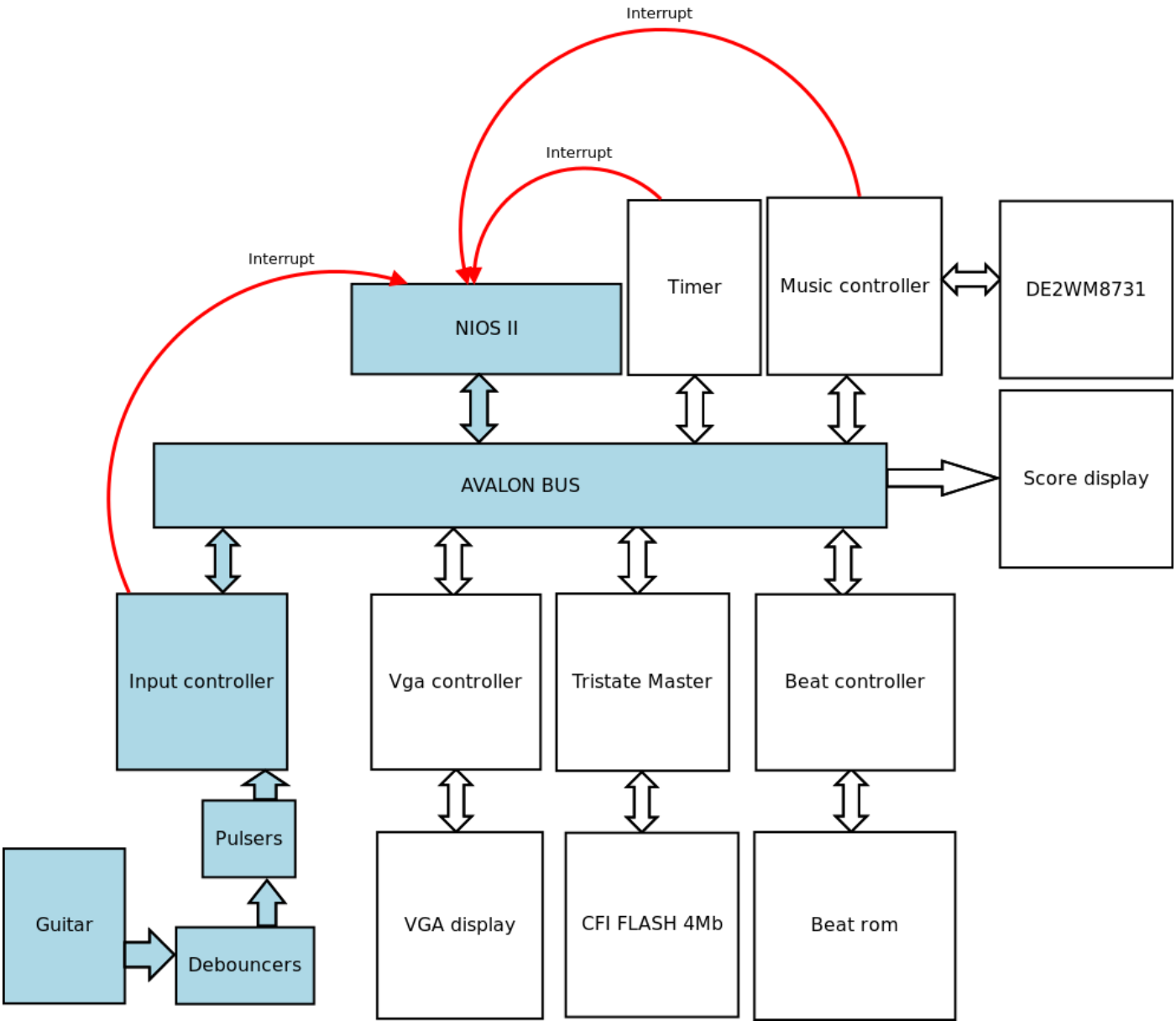
The pseudo code looks like this

```
// The callback
callback(){
      increase the address to fetch the new sample;
      should_fetch_new_sample = 1;
      WRITENOTE(bufferedsample); //prepared in the infinite loop
}
…
//Inside the main loop
for(;;){
...
if(should_fetch_new_sample == 1){
    next_sample = READNEWNOTE(notes_address);
    should_fetch_new_sample = 0;
    }
…
}
```

II.3 How we get the user input

To get the user input. We have customized a PS2 guitar controller, connected the buttons to wires and drive those wires through an ethernet cable to an intermediary board that we have designed. From this intermediary board, the information goes into the fpga using a ribbon cable. There we debounce and pulse the signal and notify the processor with an interruption when a key is pressed.

Here is a diagram summarizing all the components involved in the process:



II.3.1 The input controller

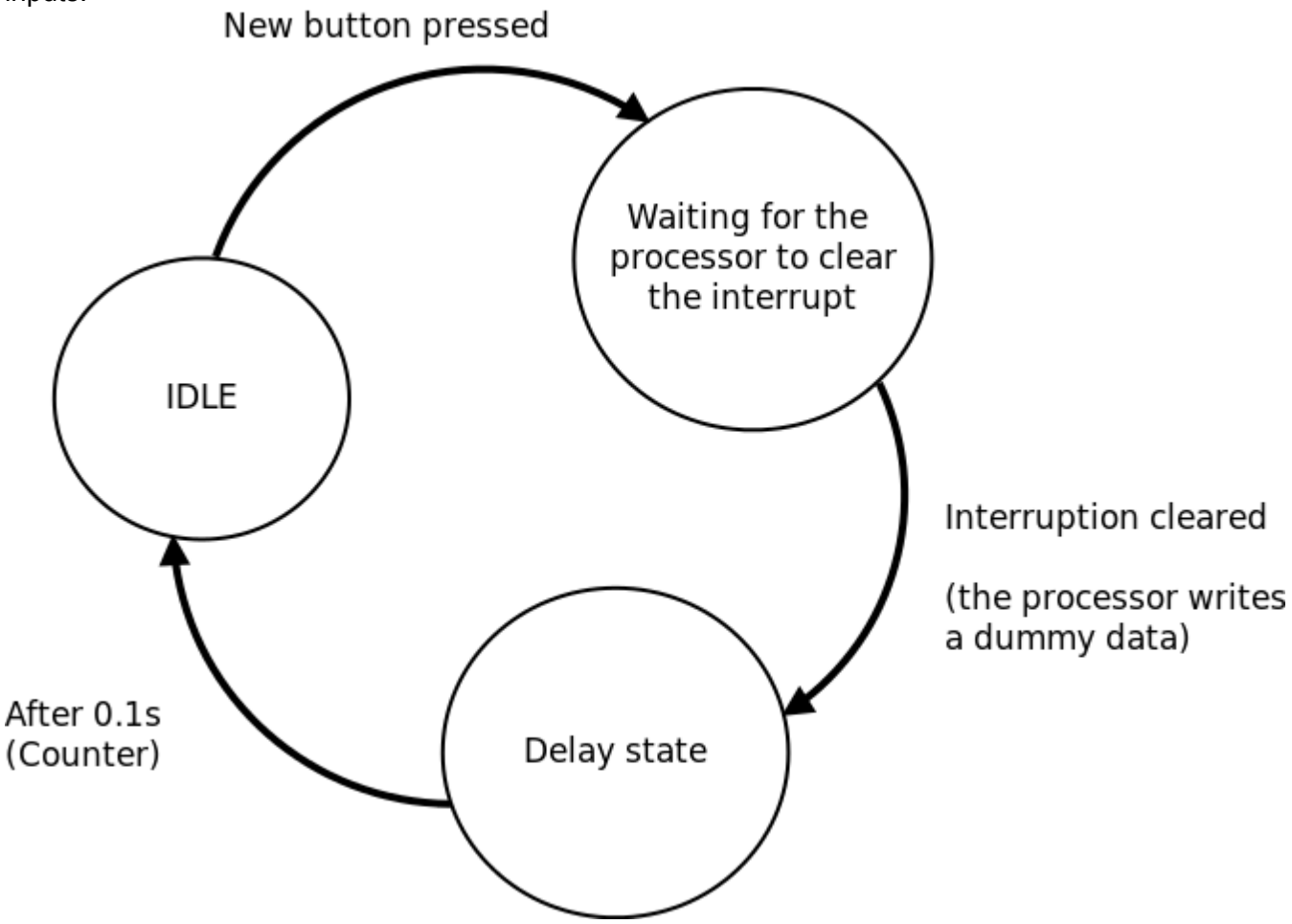
Introduction

We use the input controller to handle the key inputs on the guitar and interrupt the processor when a key is pressed.
The guitar buttons are connected to GPIOS, the corresponding signals are debounced and pulsed. Pulsed mean that the transition of the signal from 0 to 1 results in a pulse that lasts during one clock cycle. We have redesigned several times the debouncer part to obtain satisfactory result, we ended up using a design found on the internet that worked very well.

Architecture

The architecture of this module has been very hierarchical: each button's signal goes through its

own debouncer and its pulser. Then if a button is detected as pressed, an interruption is sent to the processor and the button number is made available to the processor.
Then the module waits a little and is again available to new buttons
inputs:

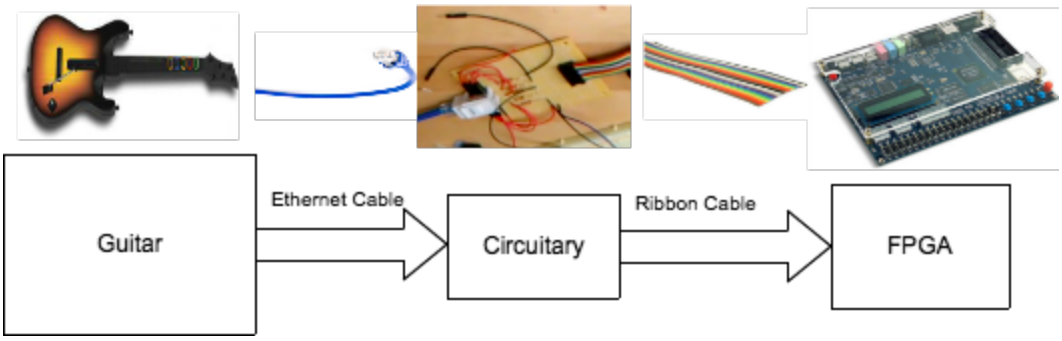


II.2.2 The Guitar

Introduction

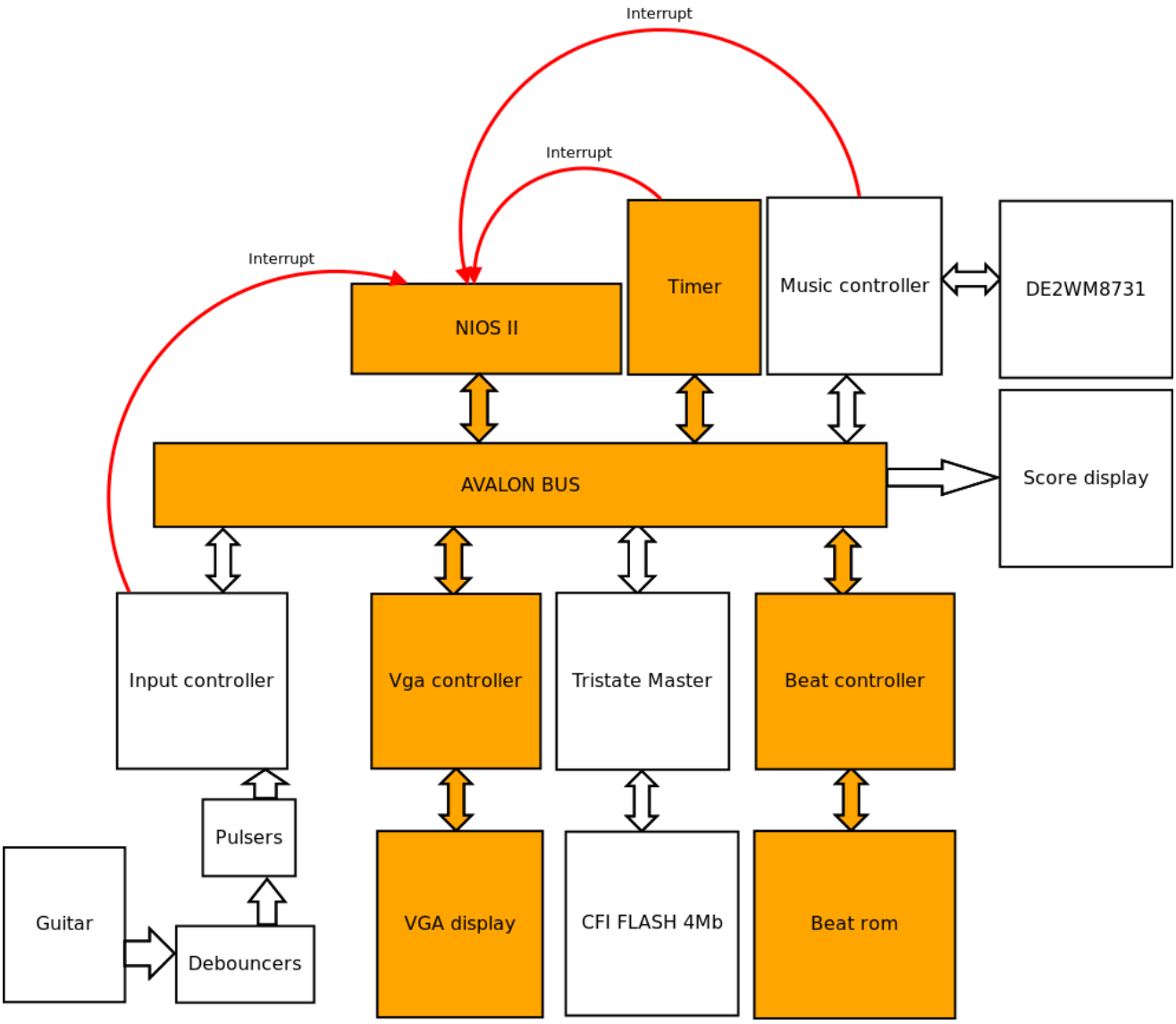
In order to make the game more natural and engrossing,instead of taking the user input through the keyboard we a guitar with 5 different color buttons. Whenver a user sees a new note on the screen ,he/she needs to press the corresponding color button of the guitar in order to fetch more points.
We thank professor for his suggestions to get a Playstation guitar and re-do its circuitry instead of building the whole guitar ourselves.

Architecture



II.3 How we dealt with the beats falling

The handling of the beats of the song is the most complex aspect of our project.
After having extracted the beats from the song, we quantify them to store them at the precision of 0.01s. We display the beats falling synchronized with the song being played.
To do so we ask a recurrent timer to interrupt the processor. When interrupted, the processor makes all the beats on the screen falling by 2 pixels and also take care of new beats on the top of the screen. The processor maps the VGA internal memory with a circular buffer to efficiently deal with the problem of finding the note matched by a user input.
Here is a diagram that highlights all the component at stakes in this aspect of the project:



II.3.1 The beat controller

Introduction

In order to build a game which was more natural and engrossing ,we wanted to have a relevant timing for displaying the notes on the screen.Instead of displaying the notes randomly ,we wanted to match their timings with the music.We used the beat tracking algorithm to compute the specific instants at which we could display a note on the screen.

Extraction

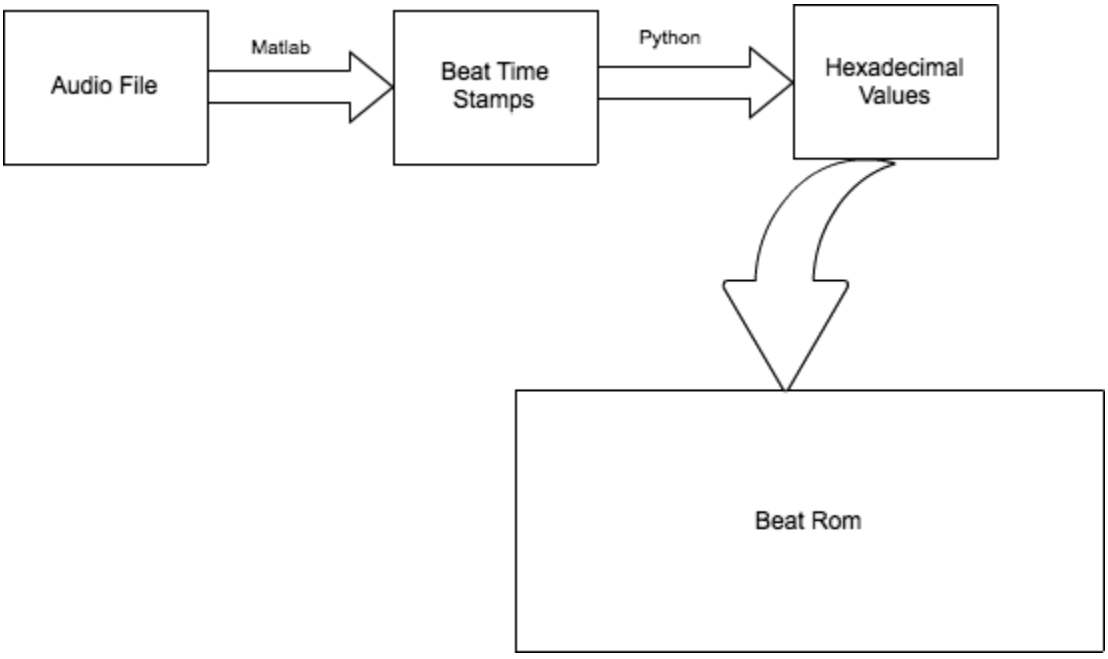
The timestamps of the beats of the song are extracted. These timestamps reflect exactly when

we need to display a new note on the screen.For beat extraction we have used the beat tracking algorithm from LABROSA(http://labrosa.ee.columbia.edu/) .The beats are extracted on Matlab and a text file is generated for the time stamps of the beats.The beats are stored at a resolution of 10^-2 second.

The beats time stamps :
RhapsodyBeats.txt
4 5 9 12 15 18 21 25 28 31 34 37 41 44 47 50 53 56 60 63 66 69 72 75 78 82 85 88 91 . . .

Architecture

Once the time stamps are extracted using Matlab they are stored at a resolution of 10^-2 seconds .These values are then converted into hexadecimal and are then stored in the RAM using the Beat Ram component.
The Beat Ram Component is instantiated in the Beat Controller.



The Beat Controller is an Avalon component .It communicates with the processor,sending it the beat time stamp whenever a request is sent by the processor.The beat controller does not drive the communication and neither send interruptions.

II.3.2 The VGA controller

Introduction

The display is one of the essential parts of the game since it is through the audio and display that the information is sent to the user.At a particular instant,the screen displays a number of notes alligned on five different vertical axis.These notes inform the user as to when and what note he needs to press on the guitar.

Architecture

The sprites consist of 5 different color button that are displayed on the different vertical axis.The images of suitable color buttons are downloaded from the internet and then passed to a Python script which extracts the hex value corresponding to every pixel.The sprite size
we are using is 32 pixels by 32 pixels.We have used five sprites for the project.

The Sprites are stored in the ROM of the FPGA.We used a single array for the storing the sprites and by using a calculated offset values we are able to read pixels of the separate Sprites individually.
The software writes on the RAM the information of where it needs to display a note.The 16 cells of the RAM are constantly read in the module VGA controller. The data for all these cells is sent to another component which returns an enable bit,the exact Sprite number and the corresponding cell indexes of the sprite from where the pixel value needs to be read.
The data corresponding to the values for which enable=1 is read from the ROM through another component.
This data read is then mapped to the Red,Green and Blue channels of the screen.

We have computed the sprites for the FPGA using the following Python script.
It allows us to transform a 32x32 image (jpg, png or gif) into a VHDL array ready for integration in the system

```
//TODO Integrate this in the global system
//TODO correct the script to actually output the right format
from PIL import Image
import sys
print "("
im = Image.open(sys.argv[1])
a = im.load()
for x in range(32):
 for y in range(32):
      if (a[x,y][3]>10):
        R,G,B,transp = a[x,y][0],a[x,y][1],a[x,y][2],0
      else:
        R,G,B,transp= 0,0,0,1
      outpixel = (transp << 30)+(R<<2)+(G<<12)+(B<<22)
```

```
outpixel = hex(outpixel)
padlength = (10-len(outpixel))
outpixel = "x\""+"0"*padlength+outpixel[2:]+"\","
if (y == 31):
  print outpixel[:-1]+"),\n("
else:
  print outpixel
```

II.3.3 The timer

Introduction

We use one central timer to synchronize the behavior of the system.
Its role is to interrupt the processor to:

- enable it to move the notes one step down
- possibly introduce a new note on the screen

We have decided to trigger the timer's interruption every 0.01 s.

Usage and design choice

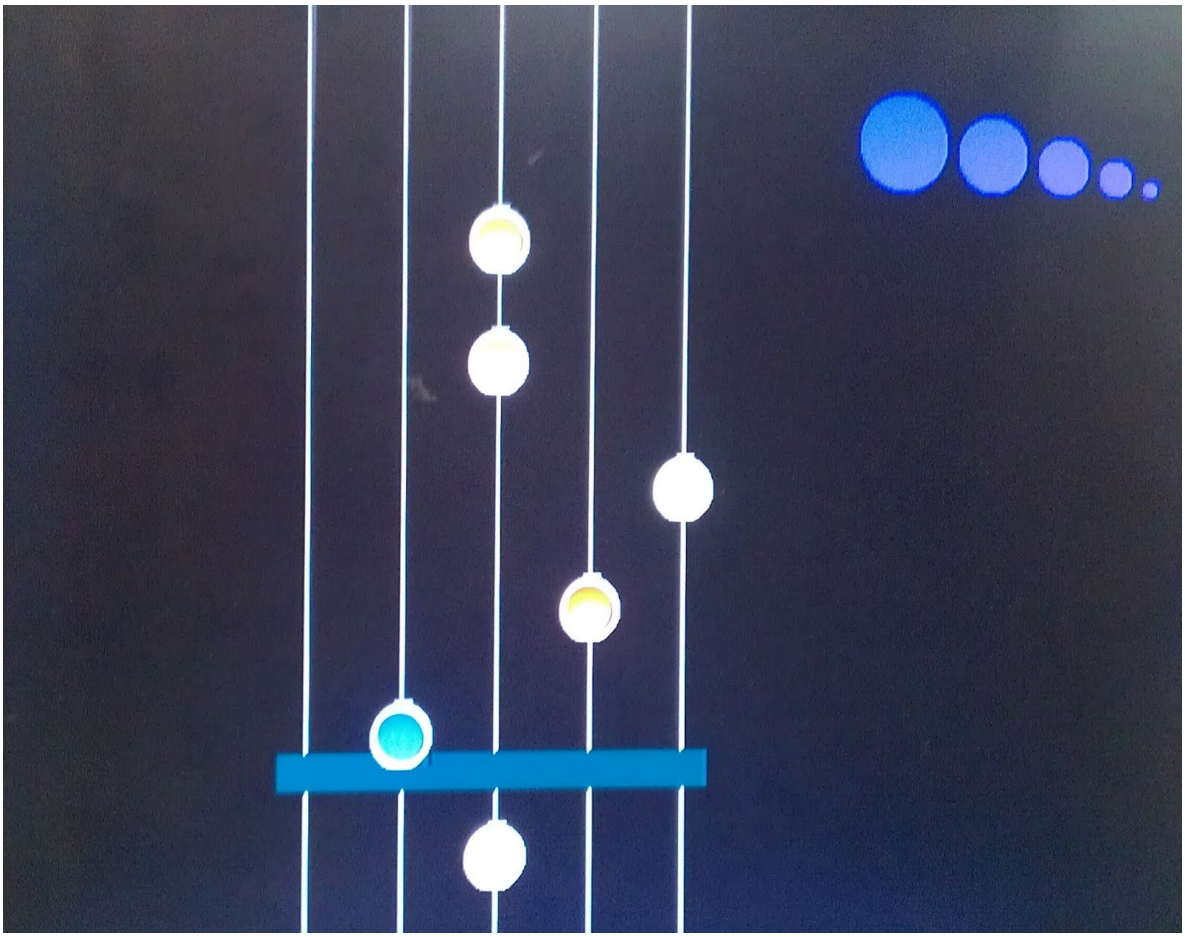
We initially decided to rely on our own timer. It was working but we then realized that there is a built in timer provided by Altera in SOPC. We are using it in the final version of our project.
We have used its datasheet to understand how to use it and we run it in **continuous mode** to avoid introducing errors that would be cumulative and detrimental to the quality of the game.

II.4 Computation and display of the score

We compute the score using the following algorithm:
- When the user presses a key:
if there is a note **(corresponding to this key)**
and **(closer than a given distance threshold to the bar of the guitar)**,
then **the note is counted and removed, the score is increased**
- Otherwise the note is not counted

This is actually the secondary score, it is displayed on the hex displayed (details provided below)

Another score is computed and represented by purple balls, here is how it works:



To get the number of balls we use this formula:
5 * (number of beats touched so far)/(total number of beats that made it+1).
The plus one is to make things easier and avoid division by zero.

This number will be 5 (balls) in case of a perfect score and 0 in case of a very bad user.
It is an intuitive way to visualize the score that the user can see in real time unlike a number.

By the way, we only start start the computation after 15 seconds to get a relevant number of balls so that the score won't fluctuate too quickly which may disorient the user.

II.4.1 Hex display

Introduction

Its role is to display the score of the ongoing game.

Architecture

Its architecture is kept simple. It contains one single 16 bits register writable by the NIOS. This register is displayed on the four rightmost hex displays of the board using four hex decoders.

II.5 Making all work together, results

To make the project working, we have basically taken the four aspects we have just described and put them together.
We have achieved our objective and have a fully functional Guitar Hero clone on a FPGA.

II.6 Project Timeline

Here is the actual timeline of our project, different colors indicates its different aspects

III Lesson learnt and role repartition

III.1 Lessons learnt

- Never leave an if statement incomplete i.e. always include an else case for cases that don't satisfy the if condition.
- Make sure not lot of data is not stored in the registers but in the memory of the board.
- Have a clear picture of whatever you plan to implement before starting coding for it.
- Divide well your project.
- Be prepared for the worst case scenario, have a backup plan for all your plans.
- Be friendly and peaceful

By the way, we are willing to make a tutorial on how to use the flash memory with the nios that could be posted on the website.

III.2 Role repartition

Avijit:

- Done most of the construction of the guitar
- Done the sprites and vga controller
- beats rom and avalon,
- Done the beat tracking algorithm,
- extracted the samples of the songs

Laurent:

- Music controller and adapted the lab3 module
- Input controller and modules inside
- Score controller
- Timers
- Software
- All the scripts to change format between data
- Took part in the construction of the guitar

Listings

Listing 1: FPGA/beatController.vhd

```
1  -- This file is part of The Awesome Guitar Game.
2  --
3  --    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
4  --    it under the terms of the GNU General Public License as published by
5  --    the Free Software Foundation, either version 3 of the License, or
6  --    (at your option) any later version.
7  --
8  --    The Awesome Guitar Game is distributed in the hope that it will be useful,
9  --    but WITHOUT ANY WARRANTY; without even the implied warranty of
10 --    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
11 --    GNU General Public License for more details.
12 --
13 --    You should have received a copy of the GNU General Public License
14 --    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
15 --
```

```
16 --
17 -##################################################################
18 -- The awesome guitar game (A clone of Guitar Hero for FPGA #
19 -##################################################################
20 -- EMBEDDED SYSTEM PROJECT
21 -- Columbia Unviersity Spring 2012
22 --
23 -- Avijit Singh Wasu -- asw2156@columbia.edu
24 -- Laurent Charignon -- lc2817@columbia.edu
25 --
26 -- Licensed under the GPL license
27 -- Have a look at the license file in the root of
28 -- the project to have more details about the license
29 --
30 -##################################################################
31 library ieee;
32 use ieee.std_logic_1164.all;
33 use ieee.numeric_std.all;
34
35 entity beatController is
36
37   port (
38     clk         : in  std_logic;
39     reset_n     : in  std_logic;
40     read        : in  std_logic;
41     write       : in  std_logic;
42     chipselect  : in  std_logic;
43     address     : in  unsigned(9 downto 0);
44     readdata    : out unsigned(15 downto 0);
45     writedata   : in  unsigned(15 downto 0)
46     );
47 end beatController;
48
49
50
51 architecture rtl of beatController is
52
53
54 signal re: std_logic;
55 signal read_in : unsigned (15 downto 0);
56 signal write_out : unsigned (15 downto 0);
57 component beatRom is
58 port
59 (
60 clk       : in std_logic;
61 en : in std_logic;
62 address : in unsigned(9 downto 0);
63 data : out unsigned(15 downto 0)
64 );
65 end component beatRom;
66 begin
67 re      <= '1' when chipselect = '1'and read ='1'  else '0' ;
68 C0: beatRom port map(clk,re,address,readdata);
69 end rtl;
```

Listing 2: FPGA/beatRom.vhd

```
-- This file is part of The Awesome Guitar Game.
--
--    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
--    it under the terms of the GNU General Public License as published by
--    the Free Software Foundation, either version 3 of the License, or
--    (at your option) any later version.
--
--    The Awesome Guitar Game is distributed in the hope that it will be useful,
--    but WITHOUT ANY WARRANTY; without even the implied warranty of
--    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
--    GNU General Public License for more details.
--
--    You should have received a copy of the GNU General Public License
--    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
--
--
-----------------------------------------------------------------------
-- The awesome guitar game (A clone of Guitar Hero for FPGA #
-----------------------------------------------------------------------
-- EMBEDDED SYSTEM PROJECT
-- Columbia Unviersity Spring 2012
--
-- Avijit Singh Wasu -- asw2156@columbia.edu
-- Laurent Charignon -- lc2817@columbia.edu
--
-- Licensed under the GPL license
-- Have a look at the license file in the root of
-- the project to have more details about the license
--
-----------------------------------------------------------------------
--16x8bit ram
--TODO check that the ram is infered
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity beatRom is
port
(
clk      : in std_logic;
en : in std_logic;
address : in unsigned(9 downto 0);
data : out unsigned(15 downto 0)
);
end beatRom;

architecture rtl of beatRom is

type concrete_rom is array (0 to 2 ** 10 -1) of unsigned (15 downto 0);
constant ROM : concrete_rom :=(x"00b4",x"00c8",x"00f0",x"0116",x"0141",x"016a",x"0193",x"01bb",x"
    01e3",x"020a",x"0232",x"025a",x"0281",x"02a8",x"02d1",x"02fa",x"0321",x"0348",x"0370",x"039a",
    x"03c1",x"03eb",x"0414",x"043a",x"0463",x"048a",x"04b3",x"04da",x"0503",x"052b",x"0553",x"057a
    ",x"05a2",x"05ca",x"05f2",x"061a",x"0642",x"066a",x"0692",x"06ba",x"06e2",x"0709",x"0731",x"
    0759",x"0781",x"07a8",x"07d0",x"07f8",x"0820",x"0848",x"0872",x"0898",x"08bd",x"08e4",x"090c",
    x"0933",x"095a",x"0981",x"09ac",x"09d7",x"0a00",x"0a25",x"0a4c",x"0a75",x"0a9c",x"0ac4",x"0aef
    ",x"0b17",x"0b40",x"0b68",x"0b8f",x"0bb6",x"0bde",x"0c06",x"0c2f",x"0c56",x"0c7e",x"0ca6",x"0
    cce",x"0cf6",x"0d1e",x"0d45",x"0d70",x"0d98",x"0dc0",x"0de4",x"0e0c",x"0e34",x"0e5c",x"0e85",x
    "0eac",x"0ed4",x"0efb",x"0f24",x"0f4b",x"0f71",x"0f98",x"0fc2",x"0feb",x"1014",x"103c",x"1064"
```

```
    ,x"108b",x"10b3",x"10db",x"1103",x"112b",x"1153",x"117b",x"11a2",x"11cb",x"11f2",x"121a",x"
    1243",x"126c",x"1292",x"12bb",x"12e2",x"130a",x"1332",x"135a",x"1382",x"13aa",x"13d1",x"13f9",
    x"1421",x"144a",x"1471",x"1499",x"14c2",x"14e8",x"1510",x"153a",x"1560",x"1588",x"15b0",x"15d9
    ",x"15ff",x"1628",x"164f",x"1677",x"169f",x"16c7",x"16ee",x"1717",x"173f",x"1768",x"178f",x"17
    b9",x"17df",x"1808",x"182e",x"1856",x"187e",x"18a6",x"18cd",x"18f6",x"191d",x"1945",x"196d",x"
    1995",x"19bd",x"19e6",x"1a0d",x"1a36",x"1a5e",x"1a85",x"1aad",x"1ad4",x"1afc",x"1b23",x"1b4c",
    x"1b73",x"1b9c",x"1bc4",x"1bec",x"1c14",x"1c3b",x"1c64",x"1c8c",x"1cb3",x"1cdb",x"1d03",x"1d2b
    ",x"1d53",x"1d78",x"1d9c",x"1dbf",x"1de2",x"1e0a",x"1e32",x"1e5b",x"1e81",x"1eaa",x"1ed4",x"1
    f01",x"1f31",x"1f5a",x"1f83",x"1faa",x"1fd1",x"1ff9",x"2022",x"204a",x"2071",x"2099",x"20c1",x
    "20e9",x"2111",x"2139",x"2160",x"2188",x"21b0",x"21d9",x"2200",x"2228",x"2250",x"2279",x"22a2"
    ,x"22ca",x"22f2",x"2318",x"2340",x"2366",x"238b",x"23b4",x"23e0",x"2407",x"2430",x"2458",x"
    2480",x"24a8",x"24d0",x"24f7",x"251e",x"2546",x"256e",x"2596",x"25bd",x"25e2",x"260c",x"2635",
    x"265d",x"2685",x"26ac",x"26d4",x"26fd",x"2725",x"274d",x"2775",x"279d",x"27c4",x"27ed",x"2814
    ",x"283c",x"2864",x"288d",x"28b4",x"28dd",x"2904",x"292d",x"2955",x"297c",x"29a3",x"29ca",x"29
    f3",x"2a1c",x"2a43",x"2a6b",x"2a91",x"2ab9",x"2ae2",x"2b0b",x"2b32",x"2b5c",x"2b82",x"2ba9",x"
    2bd0",x"2bfa",x"2c20",x"2c4a",x"2c72",x"2c9b",x"2cc1",x"2cea",x"2d10",x"2d38",x"2d61",x"2d8a",
    x"2db0",x"2dd9",x"2e00",x"2e29",x"2e50",x"2e78",x"2ea0",x"2ec8",x"2ef0",x"2f18",x"2f40",x"2f68
    ",x"2f8f",x"2fb7",x"2fde",x"3007",x"3030",x"3058",x"307e",x"30a5",x"30cd",x"30f5",x"311d",x"
    3144",x"316c",x"3194",x"31bb",x"31e4",x"320d",x"3235",x"325d",x"3286",x"32ae",x"32d7",x"32fd",
    x"3323",x"334c",x"3373",x"339c",x"33c4",x"33ec",x"3416",x"343c",x"3463",x"348c",x"34b4",x"34dc
    ",x"3504",x"352d",x"3554",x"357c",x"35a2",x"35ca",x"35f1",x"361b",x"3645",x"366a",x"3692",x"36
    ba",x"36e2",x"370a",x"3733",x"375a",x"3782",x"37ab",x"37d3",x"37f9",x"3820",x"3847",x"386e",x"
    3896",x"38c0",x"38e9",x"3912",x"393a",x"3963",x"398a",x"39b1",x"39db",x"3a02",x"3a2a",x"3a52",
    x"3a79",x"3aa2",x"3aca",x"3af2",x"3b1c",x"3b48",x"3b6c",x"3b91",x"3bb8",x"3be0",x"3c08",x"3c31
    ",x"3c59",x"3c80",x"3ca8",x"3cd0",x"3cf8",x"3d1f",x"3d49",x"3d70",x"3d96",x"3dbe",x"3de5",x"3
    e0e",x"3e36",x"3e5e",x"3e86",x"3eae",x"3ed6",x"3efe",x"3f26",x"3f4e",x"3f76",x"3f9f",x"3fc7",x
    "3fee",x"4015",x"403d",x"4065",x"408d",x"40b4",x"40dc",x"4104",x"412c",x"4154",x"417d",x"41a5"
    ,x"41cc",x"41f3",x"421b",x"4242",x"426b",x"4293",x"42bb",x"42e3",x"430b",x"4333",x"435a",x"
    4381",x"43aa",x"43d2",x"43f9",x"4422",x"444a",x"4472",x"449a",x"44c2",x"44ea",x"4512",x"4539",
    x"4562",x"458a",x"45b1",x"45d8",x"4601",x"4629",x"4654",x"467b",x"46a1",x"46c9",x"46f0",x"4718
    ",x"4740",x"4768",x"4791",x"47ba",x"47e1",x"4808",x"482f",x"4857",x"487e",x"48a7",x"48d0",x"48
    f8",x"4920",x"4947",x"496e",x"4996",x"49be",x"49e6",x"4a0c",x"4a35",x"4a5d",x"4a84",x"4aac",x"
    4ad5",x"4afd",x"4b23",x"4b49",x"4b71",x"4b98",x"4bc0",x"4be6",x"4c0d",x"4c33",x"4c5d",x"4c82",
    x"4cad",x"4cd7",others=>x"0000");
51 begin
52
53 process(clk)
54 begin
55 if(rising_edge(clk)) then
56 if(en = '1') then
57 data <= ROM(to_integer(address));
58 end if;
59 end if;
60 end process;
61
62
63 end rtl;
```

Listing 3: FPGA/counter.vhd

```
1 library ieee;
2
3 use ieee.std_logic_1164.all;
4 use ieee.numeric_std.all;
5
6 entity counter is
7
```

```
port
(
--Synchronous reset
reset : in std_logic;
clk: in std_logic;
do: out unsigned(31 downto 0);
di : in unsigned (31 downto 0)
);

end counter;


architecture ar1 of counter is
signal value : unsigned(31 downto 0);
begin
process(clk)
   begin
   if(rising_edge(clk)) then
      if( reset='1' )
      then
         value <= di;
      else
         value <= value -1;
      end if;
   end if;
   end process;
   do <= value; --copy value to the output
end ar1;
```

Listing 4: FPGA/de2wm8731audio.vhd

```
-- This file is part of The Awesome Guitar Game.
--
--    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
--    it under the terms of the GNU General Public License as published by
--    the Free Software Foundation, either version 3 of the License, or
--    (at your option) any later version.
--
--    The Awesome Guitar Game is distributed in the hope that it will be useful,
--    but WITHOUT ANY WARRANTY; without even the implied warranty of
--    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
--    GNU General Public License for more details.
--
--    You should have received a copy of the GNU General Public License
--    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
--
--
-----------------------------------------------------------------------
-- The awesome guitar game (A clone of Guitar Hero for FPGA #
-----------------------------------------------------------------------
-- EMBEDDED SYSTEM PROJECT
-- Columbia Unviersity Spring 2012
--
-- Avijit Singh Wasu -- asw2156@columbia.edu
-- Laurent Charignon -- lc2817@columbia.edu
--
-- Licensed under the GPL license
```

```
-- Have a look at the license file in the root of
-- the project to have more details about the license
--
-----------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_wm8731_audio is
port (
    clk : in std_logic;       --  Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
    reset_n : in std_logic;
    test_mode : in std_logic;       --    Audio CODEC controller test mode
    audio_request : out std_logic;  --    Audio controller request new data
    data : in unsigned(15 downto 0);


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

architecture rtl of de2_wm8731_audio is

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

    signal lrck_divider : unsigned(15 downto 0);
    signal bclk_divider : unsigned(3 downto 0);

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

    signal shift_out : unsigned(15 downto 0);

    signal sin_out     : unsigned(15 downto 0);
    signal sin_counter : unsigned(5 downto 0);

begin



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

      elsif lrck_divider = X"30c"  then

        lrck_divider <= X"0000";
```

```
85        else
86
87          lrck_divider <= lrck_divider + 1;
88        end if;
89      end if;
90    end process;
91
92    process (clk)
93    begin
94      if rising_edge(clk) then
95        if reset_n = '0' then
96          bclk_divider <= (others => '0');
97        elsif bclk_divider = X"B" or set_lrck = '1'  then
98          bclk_divider <= X"0";
99        else
100         bclk_divider <= bclk_divider + 1;
101       end if;
102     end if;
103   end process;
104
105   set_lrck <= '1' when lrck_divider = X"30c" else '0';
106
107   process (clk)
108   begin
109     if rising_edge(clk) then
110       if reset_n = '0' then
111         lrck <= '0';
112       elsif set_lrck = '1' then
113         lrck <= not lrck;
114       end if;
115     end if;
116   end process;
117
118   -- BCLK divider
119   set_bclk <= '1' when bclk_divider(3 downto 0) = "0101" else '0';
120   clr_bclk <= '1' when bclk_divider(3 downto 0) = "1011" else '0';
121
122   process (clk)
123   begin
124     if rising_edge(clk) then
125       if reset_n = '0' then
126         bclk <= '0';
127       elsif set_lrck = '1' or clr_bclk = '1' then
128         bclk <= '0';
129       elsif set_bclk = '1' then
130         bclk <= '1';
131       end if;
132     end if;
133   end process;
134
135   -- Audio data shift output
136   process (clk)
137   begin
138     if rising_edge(clk) then
139       if reset_n = '0' then
140         shift_out <= (others => '0');
141       elsif set_lrck = '1' then
142         if test_mode = '1' then
```

```
143           shift_out <= sin_out;
144         else
145           shift_out <= data;
146         end if;
147       elsif clr_bclk = '1' then
148         shift_out <= shift_out (14 downto 0) & '0';
149       end if;
150     end if;
151   end process;
152
153     -- Audio outputs
154
155     AUD_ADCLRCK  <= lrck;
156     AUD_DACLRCK  <= lrck;
157     AUD_DACDAT   <= shift_out(15);
158     AUD_BCLK     <= bclk;
159
160     -- Self test with Sin wave
161
162     process(clk)
163     begin
164       if rising_edge(clk) then
165         if reset_n = '0' then
166             sin_counter <= (others => '0');
167         elsif lrck_lat = '1' and lrck = '0'  then
168           if sin_counter = "101111" then
169             sin_counter <= "000000";
170           else
171             sin_counter <= sin_counter + 1;
172           end if;
173         end if;
174       end if;
175     end process;
176
177     process(clk)
178     begin
179       if rising_edge(clk) then
180         lrck_lat <= lrck;
181       end if;
182     end process;
183
184     process (clk)
185     begin
186       if rising_edge(clk) then
187        if lrck_lat = '1' and lrck = '0' then
188           audio_request <= '1';
189         else
190           audio_request <= '0';
191         end if;
192       end if;
193     end process;
194
195
196
197
198
199   with sin_counter select sin_out <=
200     X"0000" when "000000",
```

```
201     X"10b4" when "000001",
202     X"2120" when "000010",
203     X"30fb" when "000011",
204     X"3fff" when "000100",
205     X"4deb" when "000101",
206     X"5a81" when "000110",
207     X"658b" when "000111",
208     X"6ed9" when "001000",
209     X"7640" when "001001",
210     X"7ba2" when "001010",
211     X"7ee6" when "001011",
212     X"7fff" when "001100",
213     X"7ee6" when "001101",
214     X"7ba2" when "001110",
215     X"7640" when "001111",
216     X"6ed9" when "010000",
217     X"658b" when "010001",
218     X"5a81" when "010010",
219     X"4deb" when "010011",
220     X"3fff" when "010100",
221     X"30fb" when "010101",
222     X"2120" when "010110",
223     X"10b4" when "010111",
224     X"0000" when "011000",
225     X"ef4b" when "011001",
226     X"dee0" when "011010",
227     X"cf05" when "011011",
228     X"c001" when "011100",
229     X"b215" when "011101",
230     X"a57e" when "011110",
231     X"9a74" when "011111",
232     X"9127" when "100000",
233     X"89bf" when "100001",
234     X"845d" when "100010",
235     X"8119" when "100011",
236     X"8000" when "100100",
237     X"8119" when "100101",
238     X"845d" when "100110",
239     X"89bf" when "100111",
240     X"9127" when "101000",
241     X"9a74" when "101001",
242     X"a57e" when "101010",
243     X"b215" when "101011",
244     X"c000" when "101100",
245     X"cf05" when "101101",
246     X"dee0" when "101110",
247     X"ef4b" when "101111",
248     X"0000" when others;
249
250 end architecture;
```

Listing 5: FPGA/DebounceCounter.vhd

```
1 --19 bit counter9
2 --19 bits since it needs to count to 20 ms
3 -- 20 ms = 500 000 * tclock *2
4 -- and 2**19 = 524 288
```

```
5
6
7
8  library ieee;
9  use ieee.std_logic_1164.all;
10 use ieee.numeric_std.all;
11
12 entity DebounceCounter is
13
14 port
15 (
16 --Synchronous reset
17 reset : in std_logic;
18 clk: in std_logic;
19 do: out unsigned(19 downto 0)
20
21 );
22
23 end DebounceCounter;
24 architecture ar1 of DebounceCounter is
25 signal value : unsigned(19 downto 0);
26 begin
27 process(clk)
28 begin
29 if(rising_edge(clk)) then
30 if( reset='1' )
31 then
32 value <= (others => '0');
33 else
34 value <= value + 1;
35 end if;
36 end if;
37 end process;
38 do <= value; --copy value to the output
39 end ar1;
```

Listing 6: FPGA/debouncer.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  -- Slight modification of
4  --http://www.deathbylogic.com/2011/03/vhdl-debounce/
5
6  entity Debouncer is
7    Port (
8      CLK : in  STD_LOGIC;
9      x : in  STD_LOGIC;
10     DBx : out  STD_LOGIC;
11 reset:in STD_LOGIC
12   );
13 end Debouncer;
14
15 architecture Behavioral of Debouncer is
16   type State_Type is (S0, S1);
17   signal State : State_Type := S0;
18
19   signal DPB, SPB : STD_LOGIC;
```

```
  signal DReg : STD_LOGIC_VECTOR (7 downto 0);
begin
  process (CLK, x)
    variable SDC : integer;
    constant Delay : integer := 50000;
  begin
    if CLK'Event and CLK = '1' then
      -- Double latch input signal
      DPB <= SPB;
      SPB <= x;

      case State is
        when S0 =>
          DReg <= DReg(6 downto 0) & DPB;

          SDC := Delay;

          State <= S1;
        when S1 =>
          SDC := SDC - 1;

          if SDC = 0 then
            State <= S0;
          end if;
        when others =>
          State <= S0;
      end case;

      if DReg = X"FF" then
        DBx <= '1';
      elsif DReg = X"00" then
        DBx <= '0';
      end if;
    end if;
  end process;
end Behavioral;
```

Listing 7: FPGA/guitartop.vhd

```
-- This file is part of The Awesome Guitar Game.
--
--    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
--    it under the terms of the GNU General Public License as published by
--    the Free Software Foundation, either version 3 of the License, or
--    (at your option) any later version.
--
--    The Awesome Guitar Game is distributed in the hope that it will be useful,
--    but WITHOUT ANY WARRANTY; without even the implied warranty of
--    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
--    GNU General Public License for more details.
--
--    You should have received a copy of the GNU General Public License
--    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
--
--
--##################################################################
-- The awesome guitar game (A clone of Guitar Hero for FPGA #
```

```
--#############################################################################
-- EMBEDDED SYSTEM PROJECT
-- Columbia Unviersity Spring 2012
--
-- Avijit Singh Wasu -- asw2156@columbia.edu
-- Laurent Charignon -- lc2817@columbia.edu
--
-- Licensed under the GPL license
-- Have a look at the license file in the root of
-- the project to have more details about the license
--
--#############################################################################
--
-- DE2 top-level module
--
-- Stephen A. Edwards, Columbia University, sedwards@cs.columbia.edu
--
-- From an original by Terasic Technology, Inc.
-- (DE2_TOP.v, part of the DE2 system board CD supplied by Altera)
--

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

entity guitar_top is

  port (
   signal CLOCK_50 : in std_logic;
   HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7 -- 7-segment displays
       : out std_logic_vector(6 downto 0);         -- (active low)
   SRAM_DQ : inout std_logic_vector(15 downto 0);
   SRAM_ADDR : out std_logic_vector(17 downto 0);
   KEY : in std_logic_vector(3 downto 0);

   SRAM_UB_N,
   SRAM_LB_N,
   SRAM_WE_N,
   SRAM_CE_N,
   SRAM_OE_N : out std_logic ;
   -- FLASH
   FL_DQ : inout std_logic_vector(7 downto 0);      -- Data bus
   FL_ADDR : out std_logic_vector(21 downto 0);  -- Address bus
   FL_WE_N,                                         -- Write Enable
   FL_RST_N,                                        -- Reset
   FL_OE_N,                                         -- Output Enable
   FL_CE_N : out std_logic;                         -- Chip Enable
   -- AUDIO
   AUD_ADCLRCK  : out std_logic;        --    Audio CODEC ADC LR Clock
    AUD_ADCDAT   : in  std_logic;        --    Audio CODEC ADC Data
    AUD_DACLRCK  : out std_logic;        --    Audio CODEC DAC LR Clock
    AUD_DACDAT   : out std_logic;        --    Audio CODEC DAC Data
    AUD_BCLK     : inout std_logic ;     --    Audio CODEC Bit-Stream Clock
   AUD_XCK  : out std_logic;
    --I2C
   iCLK : in std_logic;
    iRST_N : in std_logic;
    I2C_SCLK : out std_logic;
```

```
77      I2C_SDAT : inout std_logic;
78  -- VGA output
79
80      VGA_CLK,                                -- Clock
81      VGA_HS,                                 -- H_SYNC
82      VGA_VS,                                 -- V_SYNC
83      VGA_BLANK,                              -- BLANK
84      VGA_SYNC : out std_logic;               -- SYNC
85      VGA_R,                                  -- Red[9:0]
86      VGA_G,                                  -- Green[9:0]
87      VGA_B : out std_logic_vector(9 downto 0);   -- Blue[9:0]
88
89
90     --GPIOS
91     GPIO_0,                                          -- GPIO Connection 0
92     GPIO_1 : inout std_logic_vector(35 downto 0) -- GPIO Connection 1
93      );
94
95  end guitar_top;
96
97  architecture datapath of guitar_top is
98
99  signal reset_n :std_logic;
100 signal new_res:std_logic;
101 signal audio_clock : unsigned(1 downto 0) := "00";
102 signal counter : unsigned(15 downto 0);
103
104 --entity
105   component de2_i2c_av_config is
106   port (
107     iCLK : in std_logic;
108     iRST_N : in std_logic;
109     I2C_SCLK : out std_logic;
110     I2C_SDAT : inout std_logic
111   );
112   end component;
113 begin
114
115 process (CLOCK_50)
116 begin
117 if rising_edge(CLOCK_50) then
118 if counter = x"ffff" then
119 reset_n <= '1';
120 else
121 reset_n <= '0';
122 counter <= counter + 1;
123 end if;
124 end if;
125 end process;
126
127
128 process (CLOCK_50)
129 begin
130    if rising_edge(CLOCK_50) then
131      audio_clock <= audio_clock + "1";
132    end if;
133 end process;
134
```

```
135
136
137   i2c : de2_i2c_av_config port map (
138     iCLK      => CLOCK_50,
139     iRST_n    => '1',
140     I2C_SCLK => I2C_SCLK,
141     I2C_SDAT => I2C_SDAT
142   );
143
144
145
146 nios : entity work.nios_system port map (
147    clk                                    => CLOCK_50,
148    reset_n                                => reset_n,
149    SRAM_ADDR_from_the_sram                     => SRAM_ADDR,
150    SRAM_CE_N_from_the_sram                     => SRAM_CE_N,
151    SRAM_DQ_to_and_from_the_sram                => SRAM_DQ,
152    SRAM_LB_N_from_the_sram                     => SRAM_LB_N,
153    SRAM_OE_N_from_the_sram                     => SRAM_OE_N,
154    SRAM_UB_N_from_the_sram                     => SRAM_UB_N,
155    SRAM_WE_N_from_the_sram                     => SRAM_WE_N,
156    data_to_and_from_the_cfi_flash                 => FL_DQ,
157    address_to_the_cfi_flash                    => FL_ADDR,
158    write_n_to_the_cfi_flash                    => FL_WE_N,
159    read_n_to_the_cfi_flash                     => FL_OE_N,
160    select_n_to_the_cfi_flash                   => FL_CE_N,
161    AUD_ADCDAT_to_the_musicController_inst          => AUD_ADCDAT,
162    AUD_ADCLRCK_from_the_musicController_inst       => AUD_ADCLRCK,
163    AUD_BCLK_to_and_from_the_musicController_inst   => AUD_BCLK,
164    AUD_DACDAT_from_the_musicController_inst    => AUD_DACDAT,
165    AUD_DACLRCK_from_the_musicController_inst       => AUD_DACLRCK,
166    AUX_XCK_from_the_musicController_inst       => AUD_XCK,
167    SWITCH_1_to_the_InputController_inst        => GPIO_0(1),
168    SWITCH_2_to_the_InputController_inst        => GPIO_0(3),
169    SWITCH_3_to_the_InputController_inst        => GPIO_0(5),
170    SWITCH_4_to_the_InputController_inst        => GPIO_0(7),
171    SWITCH_5_to_the_InputController_inst        => GPIO_0(9),
172    HEX_0_from_the_scoreController_inst         => HEX0,
173    HEX_1_from_the_scoreController_inst         => HEX1,
174    HEX_2_from_the_scoreController_inst         => HEX2,
175    HEX_3_from_the_scoreController_inst         => HEX3,
176    VGA_CLK_from_the_vga                    => VGA_CLK,
177     VGA_HS_from_the_vga                    => VGA_HS,
178     VGA_VS_from_the_vga                    => VGA_VS,
179     VGA_BLANK_from_the_vga                     => VGA_BLANK,
180     VGA_SYNC_from_the_vga                  => VGA_SYNC,
181     VGA_R_from_the_vga                     => VGA_R,
182     VGA_G_from_the_vga                     => VGA_G,
183     VGA_B_from_the_vga                     => VGA_B
184
185 );
186
187 -- Todo multiplex the inputs
188
189
190   FL_RST_N <= reset_n;
191
192
```

```
193 end datapath;
```

Listing 8: FPGA/hexdecoder.vhd

```
1  -- This file is part of The Awesome Guitar Game.
2  --
3  --    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
4  --    it under the terms of the GNU General Public License as published by
5  --    the Free Software Foundation, either version 3 of the License, or
6  --    (at your option) any later version.
7  --
8  --    The Awesome Guitar Game is distributed in the hope that it will be useful,
9  --    but WITHOUT ANY WARRANTY; without even the implied warranty of
10 --    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
11 --    GNU General Public License for more details.
12 --
13 --    You should have received a copy of the GNU General Public License
14 --    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
15 --
16 --
17 -----------------------------------------------------------------------
18 -- The awesome guitar game (A clone of Guitar Hero for FPGA #
19 -----------------------------------------------------------------------
20 -- EMBEDDED SYSTEM PROJECT
21 -- Columbia Unviersity Spring 2012
22 --
23 -- Avijit Singh Wasu -- asw2156@columbia.edu
24 -- Laurent Charignon -- lc2817@columbia.edu
25 --
26 -- Licensed under the GPL license
27 -- Have a look at the license file in the root of
28 -- the project to have more details about the license
29 --
30 -----------------------------------------------------------------------
31 --Hex Decoder
32
33 library ieee;
34 use ieee.std_logic_1164.all;
35 use ieee.numeric_std.all;
36
37 entity hex_decoder is
38
39
40    port
41    (
42       di: in unsigned(3 downto 0);
43       do: out std_logic_vector(6 downto 0)
44
45    );
46
47 end hex_decoder;
48
49 --Inspired by Stephen A. Edwards' slides
50 architecture ar1 of hex_decoder is
51
52 begin
53    with di select do <=
```

```
"1000000" when x"0",
"1111001" when x"1",
"0100100" when x"2",
"0110000" when x"3",
"0011001" when x"4",
"0010010" when x"5",
"0000010" when x"6",
"1111000" when x"7",
"0000000" when x"8",
"0010000" when x"9",
"0001000" when x"A",
"0000011" when x"B",
"1000110" when x"C",
"0100001" when x"D",
"0000110" when x"E",
"0001110" when x"F",
"XXXXXXX" when others;

end ar1;
```

Listing 9: FPGA/InputController.vhd

```
-- This file is part of The Awesome Guitar Game.
--
--    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
--    it under the terms of the GNU General Public License as published by
--    the Free Software Foundation, either version 3 of the License, or
--    (at your option) any later version.
--
--    The Awesome Guitar Game is distributed in the hope that it will be useful,
--    but WITHOUT ANY WARRANTY; without even the implied warranty of
--    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
--    GNU General Public License for more details.
--
--    You should have received a copy of the GNU General Public License
--    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
--
--
-----################################################################
-- The awesome guitar game (A clone of Guitar Hero for FPGA #
-----################################################################
-- EMBEDDED SYSTEM PROJECT
-- Columbia Unviersity Spring 2012
--
-- Avijit Singh Wasu -- asw2156@columbia.edu
-- Laurent Charignon -- lc2817@columbia.edu
--
-- Licensed under the GPL license
-- Have a look at the license file in the root of
-- the project to have more details about the license
--
-----################################################################
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity InputController is
```

```
36
37    port (
38      --Avalon
39      clk         : in  std_logic;
40      reset_n     : in  std_logic;
41      read        : in  std_logic;
42      write       : in  std_logic;
43      chipselect  : in  std_logic;
44      address     : in  unsigned(9 downto 0);
45      readdata    : out unsigned(15 downto 0);
46      writedata   : in  unsigned(15 downto 0);
47      inter     : out std_logic;
48      --To switches
49      SWITCH_1    : in std_logic;
50      SWITCH_2 : in std_logic;
51      SWITCH_3 : in std_logic;
52      SWITCH_4 : in std_logic;
53      SWITCH_5 : in std_logic
54      );
55  end InputController;
56
57  architecture rtl of InputController is
58
59  type states is (IDLE, PRESSED,WAITST);
60  signal state, next_state : states;
61
62  signal reset:std_logic;
63  signal counter: integer;
64  signal count_reached : std_logic;
65  signal is_pressed:std_logic;
66  signal last_pressed : unsigned (15 downto 0);
67
68  -- A bunch of signals for the button
69  -- The button clicks are first debounced and then
70  -- Pulsed
71  -- The naming follow this convention:
72  -- Initial     SWITCH_1   ______--_-_-----------_--_---------
73  -- Debounced   DSWITCH_1  ----------_----------_---------
74  -- Pulsed      PSWITCH_1  ----------_----------------------
75
76  --Switch 1
77  signal DSWITCH_1:std_logic;
78  signal NDSWITCH_1:std_logic;
79  signal PSWITCH_1:std_logic;
80  --Switch 2
81  signal DSWITCH_2:std_logic;
82  signal NDSWITCH_2:std_logic;
83  signal PSWITCH_2:std_logic;
84  --Switch 3
85  signal DSWITCH_3:std_logic;
86  signal NDSWITCH_3:std_logic;
87  signal PSWITCH_3:std_logic;
88  --Switch 4
89  signal DSWITCH_4:std_logic;
90  signal NDSWITCH_4:std_logic;
91  signal PSWITCH_4:std_logic;
92  ---Switch 5
93  signal DSWITCH_5:std_logic;
```

```
signal NDSWITCH_5:std_logic;
signal PSWITCH_5:std_logic;
-- Avalon helper
signal re:std_logic;
signal we:std_Logic;
--Debouncer
component debouncer is port(
   clk      : in std_logic;
   x  : in std_logic;
   DBx : out std_logic;
   reset : in std_logic
);
end component debouncer;
--Pulser
component pulser is port(
   clk      : in std_logic;
   key_in   : in std_logic;
   key_out : out std_logic;
   reset : in std_logic
);
end component pulser;
begin

readdata<=last_pressed;
process(clk)
begin
   if(rising_edge(clk)) then
   if( reset_n='0' )
      then
         last_pressed <= "0000000000000000";
      else
         if PSWITCH_1 ='1' then
            last_pressed <= "0000000000000001";
         elsif PSWITCH_2 ='1' then
            last_pressed <= "0000000000000010";
         elsif PSWITCH_3 ='1' then
            last_pressed <= "0000000000000100";
         elsif PSWITCH_4 ='1' then
            last_pressed <= "0000000000001000";
         elsif PSWITCH_5 ='1' then
            last_pressed <= "0000000000010000";
         else last_pressed <= last_pressed;
      end if;


      end if;


   end if ;
end process;









--Reset
reset <= not reset_n;
```

```
152 --Write enable
153 we <= '1' when chipselect ='1' and write='1' else '0';
154 --Read enable
155 re <= '1' when chipselect ='1' and read='1' else '0';
156 count_reached <= '1' when (counter > 5000000) else '0';
157
158
159
160
161
162 is_pressed <= '1' when PSWITCH_1 = '1' or PSWITCH_2 = '1' or PSWITCH_1 = '1' or PSWITCH_2 = '1' or
        PSWITCH_3 = '1' or PSWITCH_4 = '1' or PSWITCH_5 = '1' else '0';
163
164 --Debouncers
165 D1: debouncer port map(clk,SWITCH_1,DSWITCH_1,reset);
166 D2: debouncer port map(clk,SWITCH_2,DSWITCH_2,reset);
167 D3: debouncer port map(clk,SWITCH_3,DSWITCH_3,reset);
168 D4: debouncer port map(clk,SWITCH_4,DSWITCH_4,reset);
169 D5: debouncer port map(clk,SWITCH_5,DSWITCH_5,reset);
170 --Pulsers
171 P1: pulser port map(clk,NDSWITCH_1,PSWITCH_1,reset);
172 P2: pulser port map(clk,NDSWITCH_2,PSWITCH_2,reset);
173 P3: pulser port map(clk,NDSWITCH_3,PSWITCH_3,reset);
174 P4: pulser port map(clk,NDSWITCH_4,PSWITCH_4,reset);
175 P5: pulser port map(clk,NDSWITCH_5,PSWITCH_5,reset);
176
177
178 NDSWITCH_1<= not DSWITCH_1;
179 NDSWITCH_2<= not DSWITCH_2;
180 NDSWITCH_3<= not DSWITCH_3;
181 NDSWITCH_4<= not DSWITCH_4;
182 NDSWITCH_5<= not DSWITCH_5;
183
184
185
186
187
188
189 --FSM Strandard next_state => state
190 process(clk)
191 begin
192    if(rising_edge(clk)) then
193       if( reset_n='0' )
194       then
195          state <= IDLE;
196       else
197          state <= next_state;
198          if (state = PRESSED) then
199             counter <=0;
200          else counter <=counter +1;
201       end if;
202
203       end if;
204    end if;
205 end process;
206
207
208
```

```
209
210 --Combinational process
211 process (state,is_pressed,we,count_reached)
212 begin
213    --By default reset the interuption signal
214    inter <= '0';
215    next_state <= state;
216
217 case state is
218
219 --Waiting for a button click
220 when IDLE =>
221 inter <='0';
222 if ( is_pressed ='1')
223 then
224 next_state <= PRESSED;
225 end if;
226
227
228
229 --A button has been pressed
230 --Stay into this state until the interuption has been cleared
231 when PRESSED =>
232 inter <='1';
233 if (we='1')
234 then
235 next_state <= WAITST;
236 end if;
237
238 when WAITST =>
239 inter <='0';
240 if (count_reached ='1')
241 then
242 next_state <= IDLE;
243 end if;
244
245
246 end case;
247
248 end process;
249
250
251 end rtl;
```

Listing 10: FPGA/musicController.vhd

```
1 -- This file is part of The Awesome Guitar Game.
2 --
3 --    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
4 --    it under the terms of the GNU General Public License as published by
5 --    the Free Software Foundation, either version 3 of the License, or
6 --    (at your option) any later version.
7 --
8 --    The Awesome Guitar Game is distributed in the hope that it will be useful,
9 --    but WITHOUT ANY WARRANTY; without even the implied warranty of
10 --    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
11 --    GNU General Public License for more details.
```

```
12 --
13 --    You should have received a copy of the GNU General Public License
14 --    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
15 --
16 --
17 -#####################################################################
18 -- The awesome guitar game (A clone of Guitar Hero for FPGA #
19 -#####################################################################
20 -- EMBEDDED SYSTEM PROJECT
21 -- Columbia Unviersity Spring 2012
22 --
23 -- Avijit Singh Wasu -- asw2156@columbia.edu
24 -- Laurent Charignon -- lc2817@columbia.edu
25 --
26 -- Licensed under the GPL license
27 -- Have a look at the license file in the root of
28 -- the project to have more details about the license
29 --
30 -#####################################################################
31 library ieee;
32 use ieee.std_logic_1164.all;
33 use ieee.numeric_std.all;
34
35 entity musicController is
36 port (
37    -- Avalon:
38     clk        : in  std_logic;
39     reset_n    : in  std_logic;
40     read       : in  std_logic;
41     write      : in  std_logic;
42     chipselect : in  std_logic;
43     address    : in  unsigned(1 downto 0);
44     readdata   : out unsigned(7 downto 0);
45     writedata  : in  unsigned(7 downto 0);
46    -- connection to the component inside:
47     irq :      OUT STD_LOGIC;
48    AUD_ADCLRCK  : out  std_logic;   --    Audio CODEC ADC LR Clock
49    AUD_ADCDAT   : in   std_logic;   --    Audio CODEC ADC Data
50    AUD_DACLRCK  : out  std_logic;   --    Audio CODEC DAC LR Clock
51    AUD_DACDAT   : out  std_logic;   --    Audio CODEC DAC Data
52    AUD_BCLK     : inout std_logic ; --    Audio CODEC Bit-Stream Clock
53    AUX_XCK      : out std_logic  --    Audio CODEC Bit-Stream Clock
54   );
55 end  musicController;
56 architecture ar of musicController is
57
58
59  --FSM
60  type states is (IDLE,WAITING);
61  signal state, next_state : states;
62  --Audio Interface
63  signal audio_clock : unsigned(1 downto 0) := "00";
64  signal indata : unsigned (7 downto 0) ;
65 signal data_to_music : unsigned (15 downto 0);
66 signal we,inter:std_logic;
67  signal audio_request : std_logic;
68
69    component de2_wm8731_audio is port(
```

```
70            clk : in std_logic;       --  Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
71           reset_n : in std_logic;
72            test_mode : in std_logic;       --    Audio CODEC controller test mode
73            audio_request : out std_logic;  --    Audio controller request new data
74            data : in unsigned(15 downto 0);
75
76            -- Audio interface signals
77               AUD_ADCLRCK  : out  std_logic;   --    Audio CODEC ADC LR Clock
78            AUD_ADCDAT   : in   std_logic;   --    Audio CODEC ADC Data
79            AUD_DACLRCK  : out  std_logic;   --    Audio CODEC DAC LR Clock
80            AUD_DACDAT   : out  std_logic;   --    Audio CODEC DAC Data
81            AUD_BCLK     : inout std_logic  --    Audio CODEC Bit-Stream Clock
82
83
84        );
85     end component de2_wm8731_audio;
86  begin
87 we      <= '1' when chipselect = '1'and write ='1'  else '0' ;
88  data_to_music <= indata & x"00"; --Since the audio is on 16 bits, we pad the data with 0s
89
90    V1: de2_wm8731_audio port map (
91      clk => audio_clock(1),
92      reset_n => '1',
93      test_mode => '0',                    -- Don t output a sine wave
94      audio_request => audio_request,
95      data => data_to_music,
96
97      -- Audio interface signals
98      AUD_ADCLRCK  => AUD_ADCLRCK,
99      AUD_ADCDAT   => AUD_ADCDAT,
100     AUD_DACLRCK  => AUD_DACLRCK,
101     AUD_DACDAT   => AUD_DACDAT,
102     AUD_BCLK     => AUD_BCLK
103   );
104
105
106
107
108 --Audio clock generation
109  process (clk)
110  begin
111    if rising_edge(clk) then
112      audio_clock <= audio_clock + "1";
113    end if;
114 end process;
115
116 AUX_XCK<=audio_clock(1);
117
118
119 --FSM next state
120  process(clk)
121  begin
122     if(rising_edge(clk)) then
123        if( reset_n='0' )
124        then
125           state <= IDLE;
126        else
127           state <= next_state;
```

```
128         end if;
129     end if;
130 end process;
131 
132 
133 --Combinational process
134 process (state,we,audio_request)
135 begin
136     --By default reset the interuption signal
137     irq <= '0';
138     next_state <= state;
139 
140 case state is
141 
142     --Waiting for a note request
143     when IDLE =>
144     indata <=indata;
145         irq <='0';
146         if (audio_request = '1')   then
147         next_state <= WAITING;
148     end if;
149 
150 
151     --A button has been pressed
152     when WAITING =>
153     irq <='1';
154     if (we='1')
155     then
156     indata <=writedata;
157     next_state <= IDLE;
158     end if;
159     end case;
160 end process;
161 
162 
163 
164 end architecture;
```

Listing 11: FPGA/pulser.vhd

```
1  --Pulser generator
2  --When the input signal switches to 1
3  --it generates a pulse
4  
5  library ieee;
6  use ieee.std_logic_1164.all;
7  use ieee.numeric_std.all;
8  
9  
10 
11 entity pulser is
12 
13     port
14     (
15         clk       : in std_logic;
16         key_in    : in std_logic;
17         key_out : out std_logic;
```

```
18        reset : in std_logic
19
20    );
21
22 end pulser ;
23
24
25 architecture A1 of pulser is
26 type states is (ZERO, ONE_PULSE, ONE_STANDBY);
27 signal state, next_state : states;
28 begin
29
30    --FSM Strandard next_state => state
31    process(clk)
32    begin
33       if(rising_edge(clk)) then
34          if( reset='1' )
35             then
36                state <= ZERO;
37             else
38                state <= next_state;
39             end if;
40       end if;
41    end process;
42
43
44    --Computation of the next state
45    process (state,key_in)
46    begin
47    next_state <= state;
48    case state is
49       --State ZERO
50       --Signal generated 0
51       when ZERO =>
52          key_out <= '0';
53          if (key_in = '1')
54          then
55             next_state <= ONE_PULSE;
56          end if;
57
58       --State ONE_PULSE
59       --Signal generated 1
60       when ONE_PULSE =>
61          key_out <= '1';
62          next_state <= ONE_STANDBY;
63       --State ONE_STANDBY
64       --Signal generated: -
65       when ONE_STANDBY =>
66          key_out <= '0';
67          if (key_in = '0')
68          then
69             next_state <= ZERO;
70          end if;
71       end case;
72    end process;
73 end A1;
```

Listing 12: FPGA/rom123.vhd

```
1 -- This file is part of The Awesome Guitar Game.
2 --
3 --    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
4 --    it under the terms of the GNU General Public License as published by
5 --    the Free Software Foundation, either version 3 of the License, or
6 --    (at your option) any later version.
7 --
8 --    The Awesome Guitar Game is distributed in the hope that it will be useful,
9 --    but WITHOUT ANY WARRANTY; without even the implied warranty of
10 --    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
11 --    GNU General Public License for more details.
12 --
13 --    You should have received a copy of the GNU General Public License
14 --    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
15 --
16 --
17 -----------------------------------------------------------------------
18 -- The awesome guitar game (A clone of Guitar Hero for FPGA #
19 -----------------------------------------------------------------------
20 -- EMBEDDED SYSTEM PROJECT
21 -- Columbia Unviersity Spring 2012
22 --
23 -- Avijit Singh Wasu -- asw2156@columbia.edu
24 -- Laurent Charignon -- lc2817@columbia.edu
25 --
26 -- Licensed under the GPL license
27 -- Have a look at the license file in the root of
28 -- the project to have more details about the license
29 --
30 -----------------------------------------------------------------------
31 library ieee;
32 use ieee.std_logic_1164.all;
33 use ieee.numeric_std.all;
34 entity rom123 is
35 port (
36 Clk : in std_logic;
37 en : in std_logic; -- Read enable
38 addrx : in integer;
39 addry: in integer;
40 printspritenumber:in integer;
41 data : out unsigned(31 downto 0)
42 );
43 end rom123;
44
45 architecture imp of rom123 is
46 signal index:integer;
47
48
49 type rom_type is array (8192 downto 0) of unsigned( 31 downto 0);           --Rom used to store
       the sprites
50 --type Matrix_type is array (31 downto 0) of rom_type;
51 constant ROM1 : rom_type :=
52 (x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
       40000000", x"40000000", x"40000000", x"40000000", x"358d535c", x"358d635c", x"358d735c", x"358
       d635c", x"354d5358", x"348d234c", x"334cd338", x"318c6320", x"2f8be300", x"40000000", x"
       40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
```

```
    40000000", x"40000000", x"40000000", x"40000000",
53  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"378de378", x"37ce0380", x"394e5398", x"3bcee3bc", x"3dcf73dc", x"3
        ecfb3ec", x"3f0fc3f0", x"3ecfb3ec", x"3dcf83e0", x"3bcef3bc", x"380e1384", x"33ccf340", x"2
        fcbf300", x"2ccb42d4", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
54  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"384
        e1388", x"38ce3390", x"3b4ee3b8", x"3f8fe3f8", x"3fcff3fc", x"3c4ef3cc", x"360d5374", x"314
        c2324", x"2ecb92fc", x"2f8bb304", x"330ca340", x"38ce0398", x"3e0f63e8", x"3fcff3fc", x"3
        c8f23c8", x"340d1344", x"2dcb72e4", x"2a4a92b0", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
55  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"38ce338c", x"3
        a8ea3a8", x"3f4fe3f8", x"3f0fb3f4", x"328c733c", x"2308b230", x"17862148", x"110500bc", x"0
        dc4d084", x"0cc4d06c", x"0d04f070", x"0f051094", x"13c5c0ec", x"1cc781a4", x"2a0a82a8", x"38
        ce039c", x"3fcff3fc", x"3b4ed3b8", x"2fcbf300", x"2a0a82a8", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
56  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"390e4394", x"3c0f03c0", x"3
        fcff3fc", x"354d3364", x"1e8791e4", x"0f44909c", x"09850048", x"0806d08c", x"080830e8", x"
        0888f120", x"08c93134", x"08c8f120", x"084840f0", x"07c720a0", x"07c5c048", x"0ac4e040", x"15
        c64110", x"298a629c", x"3c0ee3cc", x"3fcff3fc", x"320c8324", x"294a529c", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
57  x"40000000", x"40000000", x"40000000", x"40000000", x"390e4394", x"3ccf33cc", x"3fcff3fc", x"2
        b4aa2c8", x"12c4f0f0", x"0905003c", x"080850e8", x"09cae1a8", x"0a8b91e4", x"0a8b61d4", x"0
        a8b01bc", x"0a8ab1a8", x"0a8a819c", x"0a8a6194", x"0aca418c", x"0a89f174", x"0948c118", x"06
        c63060", x"0ac4d038", x"1e4801bc", x"36cd8380", x"3fcff3fc", x"324ca328", x"27ca0284", x"
        40000000", x"40000000", x"40000000", x"40000000",
58  x"40000000", x"40000000", x"40000000", x"38ce4390", x"3c4f13c4", x"3fcff3fc", x"2749a288", x"0
        e846090", x"07c6b084", x"098b01b0", x"0a8bd1f4", x"0a8b41d0", x"0a8b21c4", x"0a8b11c0", x"0
        a8af1b8", x"0a8ac1a8", x"0a8a7198", x"0a8a418c", x"0a8a017c", x"0a89c16c", x"0a89b164", x"0
        ac9c168", x"08c820f0", x"06c4f014", x"1907114c", x"358d3368", x"3fcff3fc", x"2fcc0304", x"
        25897264", x"40000000", x"40000000", x"40000000",
59  x"40000000", x"40000000", x"388e238c", x"3acea3ac", x"3fcff3fc", x"2a4a72b8", x"0e846090", x"07
        c7c0c8", x"0a0bf1f8", x"0a4b81e4", x"0a4b41d0", x"0a8b41cc", x"0a8b31c8", x"0a8b21c4", x"0
        a8b11c0", x"0a8af1b8", x"0a8aa1a4", x"0a8a7194", x"0a8a3184", x"0a89f174", x"0a89b164", x"0
        a896154", x"0a896154", x"0a08f12c", x"06055024", x"1a875160", x"388df398", x"3ecfb3ec", x"2
        b0ad2b8", x"23890244", x"40000000", x"40000000",
60  x"40000000", x"40000000", x"394e5394", x"3fcff3fc", x"338cc34c", x"1284f0e4", x"07c790c0", x"0
        a0c2208", x"0a4b81e0", x"0a4b61d8", x"0a4b61d8", x"0a4b51d4", x"0a4b41d0", x"0a8b31cc", x"0
        a8b21c4", x"0a8b11c0", x"0a8ae1b4", x"0a8aa1a0", x"0a8a5190", x"0a8a1180", x"0a89d16c", x"0
        a89915c", x"0a894148", x"0a892140", x"0a08e128", x"06c5402c", x"22c90200", x"3e4f73f0", x"37
        ce0380", x"25496260", x"40000000", x"40000000",
61  x"40000000", x"37cde380", x"3bcf03c0", x"3e4f83e8", x"1d4711c0", x"09065094", x"094bf1f4", x"0
        a0bb1ec", x"0a0b81e4", x"0a0b81e0", x"0a4b71dc", x"0a4b61d8", x"0a4b51d4", x"0a4b41d0", x"0
        a8b31c8", x"0a8b21c4", x"0a8b01bc", x"0a8ac1ac", x"0a8a8198", x"0a8a3188", x"0a89f174", x"0
        a89b164", x"0a896150", x"0a89113c", x"0a88f134", x"08c830f4", x"0c45a078", x"30cc1308", x"3
        fcff3fc", x"2ccb42d4", x"20883218", x"40000000",
62  x"40000000", x"384e2388", x"3fcff3fc", x"304bd310", x"0ec510bc", x"080aa18c", x"0a0c0204", x"0
        a0ba1e8", x"0a0b91e8", x"0a0b91e4", x"0a0b81e0", x"0a4b71dc", x"0a4b61d8", x"0a4b51d4", x"0
        a8b41cc", x"0a8b31c8", x"0a8b11c0", x"0a8af1b8", x"0a8aa1a0", x"0a8a5190", x"0a8a1180", x"0
        a89c16c", x"0a897158", x"0a893144", x"0a88e130", x"0ac8e130", x"0646c088", x"1cc82190", x"3
        e4f63f0", x"370dc374", x"2348e240", x"40000000",
63  x"360d8368", x"39ce83a0", x"3fcff3fc", x"2147e1f4", x"0947b0f8", x"094c5210", x"09cbb1f0", x"09
        cbb1f0", x"09cbb1f0", x"0a0ba1ec", x"0a0b91e8", x"0a0b81e0", x"0a4b71dc", x"0a4b61d8", x"0
        a4b41d0", x"0a8b31cc", x"0a8b21c4", x"0a8b01bc", x"0a8ac1ac", x"0a8a7198", x"0a8a2184", x"0
        a89e170", x"0a89915c", x"0a894148", x"0a890138", x"0a88b120", x"09086104", x"0dc680b8", x"350
        d0344", x"3ecfb3ec", x"280a1288", x"40000000",
64  x"364d9368", x"3c8f23c8", x"3a8e73b0", x"16064144", x"080aa190", x"09cc220c", x"09cbd1f8", x"09
        cbd1f8", x"09cbc1f4", x"09cbb1f0", x"0a0ba1ec", x"0a0b91e4", x"0a0b81e0", x"0a4b61d8", x"0
```

```
        a4b51d4", x"0a8b41cc", x"0a8b21c8", x"0a8b11c0", x"0a8ad1b0", x"0a8a819c", x"0a8a3188", x"0
        a89f174", x"0a89a160", x"0a89514c", x"0a890138", x"0a88b124", x"0a88a11c", x"07c6e0a0", x"28
        caa26c", x"3fcff3fc", x"2d8b72e0", x"1cc741dc",
65  x"364da368", x"3e8fa3e8", x"340c7338", x"10070130", x"080c01ec", x"098be200", x"098be200", x"09
        cbe1fc", x"09cbd1f8", x"09cbc1f4", x"09cbb1f0", x"0a0b91e8", x"0a0b81e0", x"0a4b71dc", x"0
        a4b51d4", x"0a4b41cc", x"0a8b21c8", x"0a8b11c4", x"0a8ae1b4", x"0a8a91a0", x"0a8a418c", x"0
        a8a0178", x"0a89b164", x"0a896150", x"0a89113c", x"0a88c128", x"0a888118", x"06c780bc", x"1
        ec941d0", x"3fcff3fc", x"328ca32c", x"1e4791f0",
66  x"364d9368", x"3f8fe3f8", x"2ecb02d4", x"0d088168", x"084c520c", x"098bf204", x"098c0208", x"098
        bf204", x"09cbe1fc", x"09cbd1f8", x"09cbb1f0", x"0a0ba1e8", x"0a0b91e4", x"0a4b71dc", x"0
        a4b61d4", x"0a4b41d0", x"0a8b31c8", x"0a8b21c4", x"0a8af1b8", x"0a8aa1a0", x"0a8a518c", x"0
        a8a017c", x"0a89b164", x"0a896150", x"0a892140", x"0a88d12c", x"0a888114", x"0747d0d8", x"
        1848d184", x"3f8f73f4", x"360d8364", x"1f87f204",
67  x"35cd7360", x"3fcff3fc", x"2cca92b4", x"0c49b1a8", x"084c4208", x"098c0208", x"098c120c", x"098
        c0204", x"098be200", x"09cbd1f8", x"09cbb1f0", x"0a0ba1ec", x"0a0b91e4", x"0a4b71dc", x"0
        a4b61d8", x"0a4b41d0", x"0a8b31c8", x"0a8b21c4", x"0a8af1b8", x"0a8aa1a4", x"0a8a5190", x"0
        a8a017c", x"0a89b168", x"0a896154", x"0a892140", x"0a88d12c", x"0a888118", x"07c7e0e0", x"
        1588e178", x"3e4f33dc", x"378de378", x"2008120c",
68  x"350d5354", x"3f8fe3f8", x"2d4ae2c8", x"0cca61d8", x"084c2200", x"098c0204", x"098c0208", x"098
        c0204", x"09cbe200", x"09cbd1f8", x"09cbb1f0", x"0a0ba1e8", x"0a0b91e4", x"0a4b71dc", x"0
        a4b61d4", x"0a4b41d0", x"0a8b31c8", x"0a8b21c4", x"0a8af1b8", x"0a8aa1a4", x"0a8a518c", x"0
        a8a017c", x"0a89b164", x"0a896150", x"0a892140", x"0a88d12c", x"0a888114", x"07c7d0dc", x"
        16094190", x"3ecf53e8", x"374dd378", x"1f87f204",
69  x"344d1348", x"3ecfb3ec", x"304b72f4", x"0e4a41e0", x"084c11f8", x"098be200", x"098be200", x"09
        cbe200", x"09cbd1f8", x"09cbc1f4", x"09cbb1f0", x"0a0b91e8", x"0a0b81e0", x"0a4b71dc", x"0
        a4b51d4", x"0a4b41cc", x"0a8b21c8", x"0a8b11c4", x"0a8ae1b4", x"0a8a91a0", x"0a8a418c", x"0
        a8a0178", x"0a89b164", x"0a896150", x"0a89113c", x"0a88c128", x"0a887114", x"0747b0d0", x"1
        a09e1d0", x"3fcfd3fc", x"354d5358", x"1d8771e4",
70  x"32ccb330", x"3d0f43d4", x"360cc348", x"1289c1e8", x"078c01f0", x"09cbd1f8", x"09cbd1f8", x"09
        cbd1f8", x"09cbc1f4", x"09cbb1f0", x"0a0ba1ec", x"0a0b91e8", x"0a0b81e0", x"0a4b61d8", x"0
        a4b51d4", x"0a8b41cc", x"0a8b21c8", x"0a8b11c0", x"0a8ad1b0", x"0a8a819c", x"0a8a318c", x"0
        a89f174", x"0a89a160", x"0a89514c", x"0a89113c", x"0a88c124", x"0a887114", x"068790c0", x"218
        ae23c", x"3fcff3fc", x"310c5314", x"1a86b1b8",
71  x"30cc4314", x"39ce73a0", x"3c4e73b0", x"1a898214", x"078be1e8", x"09cbb1f0", x"09cbc1f4", x"09
        cbb1f0", x"09cbb1f0", x"0a0ba1ec", x"0a0b91e8", x"0a0b81e0", x"0a4b71dc", x"0a4b61d8", x"0
        a4b51d0", x"0a8b31cc", x"0a8b21c4", x"0a8b01bc", x"0a8ac1ac", x"0a8a7198", x"0a8a2184", x"0
        a89e170", x"0a89915c", x"0a894148", x"0a890138", x"0a88b124", x"0a88610c", x"07c7a0d0", x"2
        d8ca2e8", x"3fcff3fc", x"2acac2b4", x"1785f188",
72  x"2d8b62dc", x"358d6358", x"3fcfd3f8", x"270a2284", x"0a8b41e0", x"094bb1e8", x"0a0ba1ec", x"0
        a0ba1ec", x"0a0ba1e8", x"0a0b91e4", x"0a0b81e4", x"0a4b71dc", x"0a4b61d8", x"0a4b51d4", x"0
        a4b41cc", x"0a8b31c8", x"0a8b11c4", x"0a8af1b8", x"0a8aa1a4", x"0a8a5190", x"0a8a1180", x"0
        a89d16c", x"0a898158", x"0a893144", x"0a88f134", x"0a88a120", x"08c810f0", x"0f48a13c", x"3
        aceb3ac", x"3ecfa3ec", x"2348e240", x"0cc360f0",
73  x"40000000", x"314c5318", x"3dcf73dc", x"354c9340", x"140a11f8", x"074bc1d8", x"0a0b91e4", x"0
        a0b91e4", x"0a0b81e4", x"0a0b81e0", x"0a4b71dc", x"0a4b61d8", x"0a4b51d4", x"0a4b41d0", x"0
        a8b31c8", x"0a8b21c4", x"0a8b01bc", x"0a8ac1b0", x"0a8a819c", x"0a8a3188", x"0a89f178", x"0
        a89b164", x"0a896150", x"0a892140", x"0a88d12c", x"0ac8911c", x"05c790c0", x"224b224c", x"3
        fcff3fc", x"34cd3350", x"1c4711d0", x"40000000",
74  x"40000000", x"2e0b82e8", x"364da368", x"3f4f93ec", x"264a3280", x"0a4b41d8", x"090b91d8", x"0
        a0b71dc", x"0a4b71dc", x"0a4b61d8", x"0a4b61d8", x"0a4b51d4", x"0a4b41d0", x"0a8b31cc", x"0
        a8b21c8", x"0a8b11c0", x"0a8af1b8", x"0a8aa1a4", x"0a8a6190", x"0a8a1180", x"0a89d16c", x"0
        a89915c", x"0a89514c", x"0a890138", x"0a88c128", x"088820f4", x"0e089130", x"398e839c", x"3
        fcff3fc", x"2749e280", x"1685c178", x"40000000",
75  x"40000000", x"40000000", x"2f4be2fc", x"3ccf43d0", x"390d8378", x"1bca1230", x"074b71c4", x"0
        a0b61d4", x"0a4b61d4", x"0a4b51d4", x"0a4b51d0", x"0a4b41cc", x"0a8b31cc", x"0a8b21c8", x"0
        a8b21c0", x"0a8af1b8", x"0a8ab1a8", x"0a8a7198", x"0a8a3188", x"0a89f178", x"0a89b164", x"0
        a897154", x"0a893144", x"0a88e130", x"0a088114", x"0687c0d0", x"2c0c92dc", x"3fcff3fc", x"350
        d4350", x"1b8701c8", x"40000000", x"40000000",
```

```
76  x"40000000", x"40000000", x"2b8af2c0", x"320c8324", x"3fcff3fc", x"338c3328", x"168a8214", x"06
        cb51b8", x"0a0b41d0", x"0a8b41cc", x"0a8b31cc", x"0a8b31c8", x"0a8b21c4", x"0a8b11c0", x"0
        a8af1b8", x"0a8ac1ac", x"0a8a819c", x"0a8a418c", x"0a8a017c", x"0a89c16c", x"0a89815c", x"0
        a894148", x"0a890138", x"0a08a120", x"0587c0cc", x"22cb725c", x"3fcff3fc", x"3e0f83e0", x"
        2248a230", x"1645a174", x"40000000", x"40000000",
77  x"40000000", x"40000000", x"40000000", x"2b4ae2bc", x"34cd334c", x"3fcff3fc", x"32cc131c", x"17
        cab220", x"070b21ac", x"094b21bc", x"0acb21c4", x"0a8b11c4", x"0a8b11c0", x"0a8af1b8", x"0
        a8ac1a8", x"0a8a819c", x"0a8a518c", x"0a8a1180", x"0a89d170", x"0a89a160", x"0a896150", x"0
        ac92144", x"0888a114", x"064800e4", x"228b725c", x"3fcfb3fc", x"3fcff3fc", x"284a2290", x"
        18061190", x"40000000", x"40000000", x"40000000",
78  x"40000000", x"40000000", x"40000000", x"40000000", x"2acac2b4", x"354d6358", x"3fcff3fc", x"364
        ce350", x"1f8af264", x"0b4af1c0", x"06caf19c", x"098ae1b0", x"0a8ad1b0", x"0a8ab1a4", x"0
        a8a7198", x"0a8a418c", x"0a8a1180", x"0a89e170", x"0a89a164", x"0a896154", x"09090130", x"05
        c850f4", x"0d090148", x"2b4ca2d8", x"3fcfe3fc", x"3fcff3fc", x"2b0ad2b8", x"194671a4", x"
        40000000", x"40000000", x"40000000", x"40000000",
79  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"294a629c", x"330cc334", x"3
        fcff3fc", x"3d0e93bc", x"2dcc02f4", x"1a0b2238", x"0c4aa1b4", x"074a517c", x"070a216c", x"07
        ca0168", x"0809d160", x"0809a154", x"0789613c", x"06890124", x"0708d118", x"0dc95160", x"208
        b524c", x"38ce7394", x"3fcff3fc", x"3ecfb3ec", x"288a3294", x"19c681ac", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
80  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"2749d27c", x"2
        d8b72e0", x"3b4ee3b8", x"3fcff3fc", x"3d0e93bc", x"324ce334", x"260be2ac", x"1c0b323c", x"160
        ac1f8", x"144a81dc", x"148a71d4", x"174a91ec", x"1f0b5248", x"2b4ca2e0", x"39ce93a0", x"3
        fcff3fc", x"3fcff3fc", x"36cdc370", x"22c8c238", x"18863198", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
81  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
        24893254", x"27c9f284", x"2fcbf300", x"3a8ea3a8", x"3fcff3fc", x"3fcff3fc", x"3fcf83f4", x"3
        e4f03d4", x"3d4ee3c8", x"3e0f13d4", x"3f8f93f4", x"3fcff3fc", x"3fcff3fc", x"3f8fe3f8", x"370
        dc374", x"288a2290", x"1cc741d8", x"1685c178", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
82  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"2288a230", x"25896260", x"2a8ab2b0", x"304c230c", x"354d5358", x"
        384e1384", x"39ce639c", x"398e5398", x"374dd378", x"334cd338", x"2ccb42d4", x"25094258", x"1
        d4761e4", x"17860188", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
83  x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000", x"1c4731d4", x"1e87b1f8", x"20883214", x"
        21486220", x"20883214", x"1fc80208", x"1f07d1fc", x"1c8731d4", x"18c6419c", x"1384f148", x"
        40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"40000000", x"
        40000000", x"40000000", x"40000000", x"40000000",
84 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
       40000000",x"40000000",x"40000000",x"358d535c",x"358d635c",x"358d735c",x"358d635c",x"354d5358",
       x"348d234c",x"334cd338",x"318c6320",x"2f8be300",x"40000000",x"40000000",x"40000000",x"40000000
       ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
       40000000",
85 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
       40000000",x"378de378",x"37ce0380",x"394e5398",x"3bcee3bc",x"3dcf73dc",x"3ecfb3ec",x"3f0fc3f0",
       x"3ecfb3ec",x"3dcf83e0",x"3bcef3bc",x"380e1384",x"33ccf340",x"2fcbf300",x"2ccb42d4",x"40000000
       ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
       40000000",
86 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"384e1388",x"
       38ce3390",x"3b4ee3b8",x"3f8fe3f8",x"3fcff3fc",x"3c4ef3cc",x"360d5374",x"314c2324",x"2ecb92fc",
       x"2f8bb304",x"330ca340",x"38ce0398",x"3e0f63e8",x"3fcff3fc",x"3c8f23c8",x"340d1344",x"2dcb72e4
       ",x"2a4a92b0",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
       40000000",
87 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"38ce338c",x"3a8ea3a8",x"
       3f4fe3f8",x"3f0fb3f4",x"328c733c",x"1f062244",x"1883717c",x"1c027100",x"1f41e0c4",x"2081b0b0",
       x"1f41e0c4",x"1b0250f8",x"16c3415c",x"188551fc",x"2a0a82a8",x"38ce039c",x"3fcff3fc",x"3b4ed3b8
```

```
",x"2fcbf300",x"2a0a82a8",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
88 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3c0f03c0",x"3fcff3fc",x"
    354d3364",x"1a44e200",x"1c0220e0",x"2141a0b0",x"2501d0c4",x"27c200d0",x"2a0210dc",x"2a8220e0",
    x"2a0210dc",x"284200d4",x"2501d0c4",x"2201b0b4",x"1c0220e0",x"144431a0",x"298a629c",x"3c0ee3cc
    ",x"3fcff3fc",x"320c8324",x"294a529c",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
89 x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3ccf33cc",x"3fcff3fc",x"2a0992c4",x"
    1882c128",x"2141a0b0",x"284200d4",x"31827104",x"34c2a118",x"34029110",x"32428108",x"31027104",
    x"30426100",x"30426100",x"2f8260f8",x"2e4250f4",x"2a0210dc",x"22c1c0b8",x"16028108",x"1c4721f0
    ",x"36cd8380",x"3fcff3fc",x"324ca328",x"27ca0284",x"40000000",x"40000000",x"40000000",x"
    40000000",
90 x"40000000",x"40000000",x"40000000",x"38ce4390",x"3c4f13c4",x"3fcff3fc",x"24c8328c",x"1b4220e0",x"
    2441d0c0",x"31827104",x"3502d120",x"3302910c",x"3302910c",x"32428108",x"32428108",x"31827104",
    x"30426100",x"2f8260f8",x"2e8250f4",x"2d4240f0",x"2c0230e8",x"2d4240f0",x"27c200d0",x"1a8220dc
    ",x"1685e1a8",x"358d3368",x"3fcff3fc",x"2fcc0304",x"25897264",x"40000000",x"40000000",x"
    40000000",
91 x"40000000",x"40000000",x"388e238c",x"3acea3ac",x"3fcff3fc",x"2949b2b8",x"1b0230e8",x"2781f0d0",x"
    3502d120",x"34c2a118",x"3302910c",x"3302910c",x"3302910c",x"3302910c",x"32428108",x"32428108",
    x"31027104",x"30426100",x"2f0250f8",x"2e4250f4",x"2c0230e8",x"2b4220e4",x"2b4220e4",x"2a0210dc
    ",x"1ac230e8",x"1946c190",x"388df398",x"3ecfb3ec",x"2b0ad2b8",x"23890244",x"40000000",x"
    40000000",
92 x"40000000",x"40000000",x"394e5394",x"3fcff3fc",x"338cc34c",x"16c2e138",x"2701f0cc",x"3543012c",x"
    34c2a118",x"34029110",x"34029110",x"3302910c",x"3302910c",x"3302910c",x"3302910c",x"32428108",
    x"32428108",x"31027104",x"30426100",x"2e8250f4",x"2dc240f0",x"2c0230e8",x"2b4220e4",x"2b4220e4
    ",x"2a0210dc",x"1502c11c",x"22c90200",x"3e4f73f0",x"37ce0380",x"25496260",x"40000000",x"
    40000000",
93 x"40000000",x"37cde380",x"3bcf03c0",x"3e4f83e8",x"1804f1f8",x"23c1c0bc",x"3502d120",x"3502b11c",x"
    34c2a118",x"34c2a118",x"34029110",x"34029110",x"3302910c",x"3302910c",x"3302910c",x"3302910c",
    x"32428108",x"31827104",x"30426100",x"2f0250f8",x"2e4250f4",x"2c0230e8",x"2b4220e4",x"2a8220e0
    ",x"2a8220e0",x"27c200d0",x"0ec4014c",x"30cc1308",x"3fcff3fc",x"2ccb42d4",x"20883218",x"
    40000000",
94 x"40000000",x"384e2388",x"3fcff3fc",x"304bd310",x"1a428104",x"30426100",x"3543012c",x"34c2a118",x"
    34c2a118",x"34c2a118",x"34c2a118",x"34029110",x"34029110",x"3302910c",x"3302910c",x"3302910c",
    x"32428108",x"32428108",x"31027104",x"30426100",x"2e8250f4",x"2d4240f0",x"2b4220e4",x"2b4220e4
    ",x"2a0210dc",x"2a0210dc",x"1c42b11c",x"1cc82190",x"3e4f63f0",x"370dc374",x"2348e240",x"
    40000000",
95 x"360d8368",x"39ce83a0",x"3fcff3fc",x"1c060228",x"2781f0d0",x"35833138",x"3502b11c",x"3502b11c",x"
    3502b11c",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"34029110",x"3302910c",x"3302910c",
    x"3302910c",x"32428108",x"31827104",x"30426100",x"2e8250f4",x"2e4250f4",x"2c0230e8",x"2b4220e4
    ",x"2a8220e0",x"2a0210dc",x"284200d4",x"0f450164",x"350d0344",x"3ecfb3ec",x"280a1288",x"
    40000000",
96 x"364d9368",x"3c8f23c8",x"3a8e73b0",x"16c3c198",x"30426100",x"3543012c",x"3502d120",x"3502d120",x"
    3502b11c",x"3502b11c",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"3302910c",x"3302910c",
    x"3302910c",x"32428108",x"31827104",x"30426100",x"2f0250f8",x"2e4250f4",x"2c0230e8",x"2b4220e4
    ",x"2a8220e0",x"2a0210dc",x"2a0210dc",x"14c3d17c",x"28caa26c",x"3fcff3fc",x"2d8b72e0",x"1
    cc741dc",
97 x"364da368",x"3e8fa3e8",x"340c7338",x"1d831148",x"3542e124",x"3502d120",x"3502d120",x"3502d120",x"
    3502d120",x"3502b11c",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"3302910c",x"3302910c",
    x"3302910c",x"32c2810c",x"32428108",x"31027104",x"2f8260f8",x"2e4250f4",x"2c0230e8",x"2b4220e4
    ",x"2a8220e0",x"2a0210dc",x"290210d8",x"1cc2f140",x"1ec941d0",x"3fcff3fc",x"328ca32c",x"1
    e4791f0",
98 x"364d9368",x"3f8fe3f8",x"2ecb02d4",x"2602a110",x"35833138",x"3542e124",x"3543012c",x"3542e124",x"
    3502d120",x"3502d120",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"33829110",x"3302910c",
    x"3302910c",x"3302910c",x"32428108",x"31027104",x"30426100",x"2e8250f4",x"2c0230e8",x"2b4220e4
    ",x"2b4220e4",x"2a0210dc",x"290210d8",x"234260fc",x"1848d184",x"3f8f73f4",x"360d8364",x"1
    f87f204",
99 x"35cd7360",x"3fcff3fc",x"2cca92b4",x"2e4250f4",x"35833138",x"3543012c",x"3543012c",x"3543012c",x"
```

```
3502d120",x"3502d120",x"3502b11c",x"3502b11c",x"34c2a118",x"34029110",x"34029110",x"3302910c",
x"3302910c",x"3302910c",x"32428108",x"31027104",x"30426100",x"2e8250f4",x"2cc240ec",x"2b4220e4
",x"2b4220e4",x"2a0210dc",x"290210d8",x"2781f0d0",x"1588e178",x"3e4f33dc",x"378de378",x"
2008120c",
x"350d5354",x"3f8fe3f8",x"2d4ae2c8",x"3082910c",x"3543012c",x"3543012c",x"3543012c",x"3543012c",x"
3502d120",x"3502d120",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"33829110",x"3302910c",
x"3302910c",x"3302910c",x"32428108",x"31027104",x"30426100",x"2e8250f4",x"2c0230e8",x"2b4220e4
",x"2b4220e4",x"2a0210dc",x"290210d8",x"268200d4",x"16094190",x"3ecf53e8",x"374dd378",x"1
f87f204",
x"344d1348",x"3ecfb3ec",x"304b72f4",x"2cc30138",x"3543012c",x"3502d120",x"3502d120",x"3502d120",x"
3502d120",x"3502b11c",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"3302910c",x"3302910c",
x"3302910c",x"32c2810c",x"32428108",x"31027104",x"2f8260f8",x"2e4250f4",x"2c0230e8",x"2b4220e4
",x"2a8220e0",x"2a0210dc",x"28c200d8",x"230260fc",x"1a09e1d0",x"3fcfd3fc",x"354d5358",x"1
d8771e4",
x"32ccb330",x"3d0f43d4",x"360cc348",x"2603d188",x"3542e124",x"3502d120",x"3502d120",x"3502d120",x"
3502b11c",x"3502b11c",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"3302910c",x"3302910c",
x"3302910c",x"32428108",x"31827104",x"30426100",x"2f8260f8",x"2e4250f4",x"2c0230e8",x"2b4220e4
",x"2a8220e0",x"2a0210dc",x"28c200d8",x"1cc2f140",x"218ae23c",x"3fcff3fc",x"310c5314",x"1
a86b1b8",
x"30cc4314",x"39ce73a0",x"3c4e73b0",x"20c52200",x"3502b11c",x"3502b11c",x"3502b11c",x"3502b11c",x"
3502b11c",x"3502b11c",x"34c2a118",x"34c2a118",x"34029110",x"34029110",x"3302910c",x"3302910c",
x"3302910c",x"32428108",x"31827104",x"30426100",x"2e8250f4",x"2e4250f4",x"2c0230e8",x"2b4220e4
",x"2a8220e0",x"2a0210dc",x"28c200d8",x"16040198",x"2d8ca2e8",x"3fcff3fc",x"2acac2b4",x"1785
f188",
x"2d8b62dc",x"358d6358",x"3fcfd3f8",x"2587e284",x"33829110",x"34c2a118",x"3502b11c",x"3502b11c",x"
34c2a118",x"34c2a118",x"34c2a118",x"34029110",x"34029110",x"3302910c",x"3302910c",x"3302910c",
x"32c2810c",x"32428108",x"31027104",x"30426100",x"2e8250f4",x"2dc240f0",x"2b8230e4",x"2b4220e4
",x"2a8220e0",x"2a0210dc",x"27c200d0",x"138651d0",x"3aceb3ac",x"3ecfa3ec",x"2348e240",x"0
cc360f0",
x"40000000",x"314c5318",x"3dcf73dc",x"354c9340",x"2843b178",x"3502b11c",x"34c2a118",x"34c2a118",x"
34c2a118",x"34c2a118",x"34029110",x"34029110",x"3302910c",x"3302910c",x"3302910c",x"3302910c",
x"32428108",x"31827104",x"30426100",x"2f0250f8",x"2e4250f4",x"2c0230e8",x"2b4220e4",x"2b4220e4
",x"2a0210dc",x"290210d8",x"1dc2d12c",x"224b224c",x"3fcff3fc",x"34cd3350",x"1c4711d0",x"
40000000",
x"40000000",x"2e0b82e8",x"364da368",x"3f4f93ec",x"25872268",x"3302910c",x"34c2a118",x"34029110",x"
34029110",x"34029110",x"34029110",x"3302910c",x"3302910c",x"3302910c",x"3302910c",x"32428108",
x"32428108",x"31027104",x"30426100",x"2e8250f4",x"2dc240f0",x"2c0230e8",x"2b4220e4",x"2a8220e0
",x"2a0210dc",x"27c200d0",x"14c571dc",x"398e839c",x"3fcff3fc",x"2749e280",x"1685c178",x"
40000000",
x"40000000",x"40000000",x"2f4be2fc",x"3ccf43d0",x"390d8378",x"2744a1cc",x"33829110",x"33829110",x"
33829110",x"3302910c",x"3302910c",x"3302910c",x"3302910c",x"3302910c",x"32c2810c",x"32428108",
x"31027104",x"30426100",x"2f0250f8",x"2e4250f4",x"2c0230e8",x"2b4220e4",x"2b4220e4",x"2a0210dc
",x"28c200d8",x"1a83616c",x"2c0c92dc",x"3fcff3fc",x"350d4350",x"1b8701c8",x"40000000",x"
40000000",
x"40000000",x"40000000",x"2b8af2c0",x"320c8324",x"3fcff3fc",x"324b9320",x"2cc39168",x"32c2810c",x"
3302910c",x"3302910c",x"3302910c",x"3302910c",x"3302910c",x"32428108",x"32428108",x"31827104",
x"30426100",x"2f8260f8",x"2e8250f4",x"2d4240f0",x"2b8230e4",x"2b4220e4",x"2a8220e0",x"290210d8
",x"2042a11c",x"238a9270",x"3fcff3fc",x"3e0f83e0",x"2248a230",x"1645a174",x"40000000",x"
40000000",
x"40000000",x"40000000",x"40000000",x"2b4ae2bc",x"34cd334c",x"3fcff3fc",x"308a9300",x"2e839164",x"
32428108",x"32428108",x"3302910c",x"32c2810c",x"32428108",x"32428108",x"31827104",x"30426100",
x"30426100",x"2e8250f4",x"2dc240f0",x"2c0230e8",x"2b4220e4",x"2b4220e4",x"290210d8",x"2202a118
",x"24c9827c",x"3fcfb3fc",x"3fcff3fc",x"284a2290",x"18061190",x"40000000",x"40000000",x"
40000000",
x"40000000",x"40000000",x"40000000",x"40000000",x"2acac2b4",x"354d6358",x"3fcff3fc",x"34cc033c",x"
2cc4b1c0",x"32428108",x"31027104",x"31827104",x"31827104",x"31027104",x"30426100",x"2f8260f8",
x"2e8250f4",x"2e4250f4",x"2c0230e8",x"2b4220e4",x"2a0210dc",x"284200d4",x"1fc39184",x"2bcb92dc
",x"3fcfe3fc",x"3fcff3fc",x"2b0ad2b8",x"194671a4",x"40000000",x"40000000",x"40000000",x"
```

```
      40000000",
111 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"294a629c",x"330cc334",x"3fcff3fc",x"
      3d0e93bc",x"2dc86298",x"30c3a168",x"31027104",x"2f8260f8",x"2e4250f4",x"2e4250f4",x"2cc240ec",
      x"2c0230e8",x"2b4220e4",x"2a0210dc",x"2a0210dc",x"2602e130",x"2747124c",x"38ce7394",x"3fcff3fc
      ",x"3ecfb3ec",x"288a3294",x"19c681ac",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
112 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"2749d27c",x"2d8b72e0",x"
      3b4ee3b8",x"3fcff3fc",x"3d0e93bc",x"314a22e4",x"2e067224",x"3004118c",x"3142f12c",x"3182910c",
      x"2f42d124",x"2bc3b174",x"2b0591fc",x"2d8992bc",x"39ce93a0",x"3fcff3fc",x"3fcff3fc",x"36cdc370
      ",x"22c8c238",x"18863198",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
113 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"24893254",x"
      27c9f284",x"2fcbf300",x"3a8ea3a8",x"3fcff3fc",x"3fcff3fc",x"3fcf83f4",x"3e4f03d4",x"3d4ee3c8",
      x"3e0f13d4",x"3f8f93f4",x"3fcff3fc",x"3fcff3fc",x"3f8fe3f8",x"370dc374",x"288a2290",x"1cc741d8
      ",x"1685c178",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
114 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"2288a230",x"25896260",x"2a8ab2b0",x"304c230c",x"354d5358",x"384e1384",x"39ce639c",
      x"398e5398",x"374dd378",x"334cd338",x"2ccb42d4",x"25094258",x"1d4761e4",x"17860188",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
115 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"40000000",x"40000000",x"1c4731d4",x"1e87b1f8",x"20883214",x"21486220",x"20883214",
      x"1fc80208",x"1f07d1fc",x"1c8731d4",x"18c6419c",x"1384f148",x"40000000",x"40000000",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
116 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"40000000",x"40000000",x"358d535c",x"358d635c",x"358d735c",x"358d635c",x"354d5358",
      x"348d234c",x"334cd338",x"318c6320",x"2f8be300",x"40000000",x"40000000",x"40000000",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
117 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"378de378",x"37ce0380",x"394e5398",x"3bcee3bc",x"3dcf73dc",x"3ecfb3ec",x"3f0fc3f0",
      x"3ecfb3ec",x"3dcf83e0",x"3bcef3bc",x"380e1384",x"33ccf340",x"2fcbf300",x"2ccb42d4",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
118 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"384e1388",x"
      38ce3390",x"3b4ee3b8",x"3f8fe3f8",x"3fcff3fc",x"3c4ef3cc",x"360d5374",x"314c2324",x"2ecb92fc",
      x"2f8bb304",x"330ca340",x"38ce0398",x"3e0f63e8",x"3fcff3fc",x"3c8f23c8",x"340d1344",x"2dcb72e4
      ",x"2a4a92b0",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
119 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"38ce338c",x"3a8ea3a8",x"
      3f4fe3f8",x"3f0fb3f4",x"328c733c",x"198812ec",x"098652bc",x"06c62284",x"0646627c",x"06467278",
      x"06465274",x"0685f278",x"08c5f280",x"1306f290",x"2a0a82a8",x"38ce039c",x"3fcff3fc",x"3b4ed3b8
      ",x"2fcbf300",x"2a0a82a8",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
120 x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3c0f03c0",x"3fcff3fc",x"
      354d3364",x"11c712e0",x"06060270",x"06468278",x"06c732c0",x"0787c2f4",x"07c83320",x"07c84328",
      x"07c83320",x"0787d2fc",x"070742c4",x"0646b290",x"0606026c",x"0c461254",x"298a629c",x"3c0ee3cc
      ",x"3fcff3fc",x"320c8324",x"294a529c",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
121 x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3ccf33cc",x"3fcff3fc",x"28ca62f4",x"
      0745e27c",x"06468278",x"0787d2fc",x"0e09b37c",x"138a638c",x"124a3388",x"0fc9e380",x"0d89a37c",
      x"0c497378",x"0bc96378",x"0b094374",x"0888f36c",x"07c82318",x"0686e2a4",x"06457234",x"1b47b214
      ",x"36cd8380",x"3fcff3fc",x"324ca328",x"27ca0284",x"40000000",x"40000000",x"40000000",x"
      40000000",
122 x"40000000",x"40000000",x"38ce4390",x"3c4f13c4",x"3fcff3fc",x"22c952dc",x"0605f26c",x"
      06c722b8",x"0e89b37c",x"154a9390",x"110a1384",x"110a1384",x"0fc9e380",x"0fc9e380",x"0e09b37c",
```

```
x"0c497378",x"0b094374",x"09491370",x"0888d35c",x"08489348",x"0888d35c",x"0787c2f4",x"0605d258
",x"1486c1dc",x"358d3368",x"3fcff3fc",x"2fcc0304",x"25897264",x"40000000",x"40000000",x"
40000000",
x"40000000",x"40000000",x"388e238c",x"3acea3ac",x"3fcff3fc",x"288a42d8",x"0645e264",x"0747a2e8",x"
154a9390",x"134a5388",x"110a1384",x"110a1384",x"110a1384",x"110a1384",x"0fc9e380",x"0fc9e380",
x"0d89a37c",x"0c497378",x"0a093374",x"0888f36c",x"08489348",x"08086334",x"08086334",x"07c83320
",x"0605e264",x"188721a0",x"388df398",x"3ecfb3ec",x"2b0ad2b8",x"23890244",x"40000000",x"
40000000",
x"40000000",x"40000000",x"394e5394",x"3fcff3fc",x"338cc34c",x"07c5b26c",x"074792e0",x"168ac394",x"
134a5388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",x"110a1384",x"110a1384",x"0fc9e380",
x"0f49d380",x"0d89a37c",x"0bc96378",x"09c92370",x"0888e364",x"08489348",x"0808532c",x"0808532c
",x"07c82318",x"06058234",x"22c90200",x"3e4f73f0",x"37ce0380",x"25496260",x"40000000",x"
40000000",
x"40000000",x"37cde380",x"3bcf03c0",x"3e4f83e8",x"1246a2a0",x"06c712b0",x"154a9390",x"148a838c",x"
134a5388",x"134a5388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",x"110a1384",x"110a1384",
x"0fc9e380",x"0e09b37c",x"0c497378",x"0a093374",x"0888f36c",x"08489348",x"08086334",x"07c84328
",x"07c84328",x"0787c2f4",x"08c561b4",x"30cc1308",x"3fcff3fc",x"2ccb42d4",x"20883218",x"
40000000",
x"40000000",x"384e2388",x"3fcff3fc",x"304bd310",x"06c60278",x"0c497378",x"168ac394",x"138a638c",x"
134a5388",x"134a5388",x"134a5388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",x"110a1384",
x"0fc9e380",x"0fc9e380",x"0d89a37c",x"0bc96378",x"09c92370",x"0888d35c",x"08086334",x"0808532c
",x"07c83320",x"07c83320",x"06c682b0",x"1cc82190",x"3e4f63f0",x"370dc374",x"2348e240",x"
40000000",
x"360d8368",x"39ce83a0",x"3fcff3fc",x"18c75298",x"0747a2e8",x"180af394",x"148a838c",x"148a838c",x"
148a838c",x"148a838c",x"134a5388",x"134a5388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",
x"110a1384",x"0fc9e380",x"0e09b37c",x"0c497378",x"09c92370",x"0888f36c",x"08489348",x"0808532c
",x"07c84328",x"07c82318",x"0787d2fc",x"0b0621a4",x"350d0344",x"3ecfb3ec",x"280a1288",x"
40000000",
x"364d9368",x"3c8f23c8",x"3a8e73b0",x"0a8632a0",x"0c497378",x"168ac394",x"150a838c",x"150a838c",x"
148a838c",x"148a838c",x"148a838c",x"134a5388",x"134a5388",x"124a3388",x"110a1384",x"110a1384",
x"110a1384",x"0fc9e380",x"0e09b37c",x"0c497378",x"0a093374",x"0888f36c",x"08489348",x"0808532c
",x"07c84328",x"07c82318",x"07c82318",x"0746426c",x"28caa26c",x"3fcff3fc",x"2d8b72e0",x"1
cc741dc",
x"364da368",x"3e8fa3e8",x"340c7338",x"0806e2f0",x"164ab390",x"154a9390",x"154a9390",x"154a9390",x"
150a838c",x"148a838c",x"148a838c",x"134a5388",x"134a5388",x"124a3388",x"110a1384",x"110a1384",
x"110a1384",x"108a0384",x"0f49d380",x"0d09937c",x"0b094374",x"08c90370",x"08489348",x"08086334
",x"07c84328",x"07c82318",x"07c80310",x"0746c2d8",x"1ec941d0",x"3fcff3fc",x"328ca32c",x"1
e4791f0",
x"364d9368",x"3f8fe3f8",x"2ecb02d4",x"0807d330",x"180af394",x"164ab390",x"168ac394",x"164ab390",x"
154a9390",x"150a838c",x"148a838c",x"138a638c",x"134a5388",x"124a3388",x"118a2384",x"110a1384",
x"110a1384",x"110a1384",x"0fc9e380",x"0d89a37c",x"0bc96378",x"09491370",x"08489348",x"08086334
",x"0808532c",x"07c82318",x"07c80310",x"078742f4",x"1848d184",x"3f8f73f4",x"360d8364",x"1
f87f204",
x"35cd7360",x"3fcff3fc",x"2cca92b4",x"0888f36c",x"180af394",x"168ac394",x"168ac394",x"168ac394",x"
154a9390",x"150a838c",x"148a838c",x"148a838c",x"134a5388",x"124a3388",x"124a3388",x"110a1384",
x"110a1384",x"110a1384",x"0fc9e380",x"0d89a37c",x"0bc96378",x"09491370",x"0848b350",x"08086334
",x"0808532c",x"07c82318",x"07c80310",x"0747b2f0",x"1588e178",x"3e4f33dc",x"378de378",x"
2008120c",
x"350d5354",x"3f8fe3f8",x"2d4ae2c8",x"0d89937c",x"168ac394",x"168ac394",x"168ac394",x"168ac394",x"
154a9390",x"150a838c",x"148a838c",x"138a638c",x"134a5388",x"124a3388",x"118a2384",x"110a1384",
x"110a1384",x"110a1384",x"0fc9e380",x"0d89a37c",x"0bc96378",x"09491370",x"08489348",x"08086334
",x"0808532c",x"07c82318",x"07c80310",x"074792f0",x"16094190",x"3ecf53e8",x"374dd378",x"1
f87f204",
x"344d1348",x"3ecfb3ec",x"304b72f4",x"0ec9437c",x"168ac394",x"154a9390",x"154a9390",x"154a9390",x"
150a838c",x"148a838c",x"148a838c",x"134a5388",x"134a5388",x"124a3388",x"110a1384",x"110a1384",
x"110a1384",x"108a0384",x"0f49d380",x"0d09937c",x"0b094374",x"08c90370",x"08489348",x"08086334
",x"07c84328",x"07c82318",x"0787f30c",x"074732ec",x"1a09e1d0",x"3fcfd3fc",x"354d5358",x"1
d8771e4",
```

```
134 x"32ccb330",x"3d0f43d4",x"360cc348",x"0f48a368",x"164ab390",x"150a838c",x"150a838c",x"150a838c",x"
        148a838c",x"148a838c",x"148a838c",x"134a5388",x"134a5388",x"124a3388",x"110a1384",x"110a1384",
        x"110a1384",x"0fc9e380",x"0e09b37c",x"0c497378",x"0a894374",x"0888f36c",x"08489348",x"0808532c
        ",x"07c84328",x"07c82318",x"0787f30c",x"0746c2d8",x"218ae23c",x"3fcff3fc",x"310c5314",x"1
        a86b1b8",
135 x"30cc4314",x"39ce73a0",x"3c4e73b0",x"13888338",x"148a838c",x"148a838c",x"148a838c",x"148a838c",x"
        148a838c",x"148a838c",x"134a5388",x"134a5388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",
        x"110a1384",x"0fc9e380",x"0e09b37c",x"0c497378",x"09c92370",x"0888f36c",x"08489348",x"0808532c
        ",x"07c84328",x"07c82318",x"0787e304",x"0786b2a0",x"2d8ca2e8",x"3fcff3fc",x"2acac2b4",x"1785
        f188",
136 x"2d8b62dc",x"358d6358",x"3fcfd3f8",x"21c9a2fc",x"118a2384",x"138a638c",x"148a838c",x"148a838c",x"
        138a638c",x"134a5388",x"134a5388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",x"110a1384",
        x"108a0384",x"0fc9e380",x"0d89a37c",x"0bc96378",x"09c92370",x"0888e364",x"0808733c",x"0808532c
        ",x"07c84328",x"07c82318",x"0787c2f4",x"0e07b224",x"3aceb3ac",x"3ecfa3ec",x"2348e240",x"0
        cc360f0",
137 x"40000000",x"314c5318",x"3dcf73dc",x"354c9340",x"10890374",x"148a838c",x"134a5388",x"134a5388",x"
        134a5388",x"134a5388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",x"110a1384",x"110a1384",
        x"0fc9e380",x"0e09b37c",x"0c497378",x"0a093374",x"0888f36c",x"08489348",x"08086334",x"0808532c
        ",x"07c82318",x"07c80310",x"0746d2dc",x"224b224c",x"3fcff3fc",x"34cd3350",x"1c4711d0",x"
        40000000",
138 x"40000000",x"2e0b82e8",x"364da368",x"3f4f93ec",x"1f49a324",x"110a1384",x"134a5388",x"124a3388",x"
        124a3388",x"124a3388",x"124a3388",x"110a1384",x"110a1384",x"110a1384",x"110a1384",x"0fc9e380",
        x"0fc9e380",x"0d89a37c",x"0bc96378",x"09c92370",x"0888e364",x"08489348",x"0808532c",x"07c84328
        ",x"07c82318",x"0787c2f4",x"0c476270",x"398e839c",x"3fcff3fc",x"2749e280",x"1685c178",x"
        40000000",
139 x"40000000",x"40000000",x"2f4be2fc",x"3ccf43d0",x"390d8378",x"15493368",x"118a2384",x"118a2384",x"
        118a2384",x"110a1384",x"110a1384",x"110a1384",x"110a1384",x"110a1384",x"108a0384",x"0fc9e380",
        x"0d89a37c",x"0c497378",x"0a093374",x"0888f36c",x"08489348",x"08086334",x"0808532c",x"07c83320
        ",x"0787f30c",x"0786c2d0",x"2c0c92dc",x"3fcff3fc",x"350d4350",x"1b8701c8",x"40000000",x"
        40000000",
140 x"40000000",x"40000000",x"2b8af2c0",x"320c8324",x"3fcff3fc",x"320c2340",x"13499380",x"108a0384",x"
        110a1384",x"110a1384",x"110a1384",x"110a1384",x"110a1384",x"0fc9e380",x"0fc9e380",x"0e09b37c",
        x"0c497378",x"0b094374",x"09491370",x"0888d35c",x"0808733c",x"0808532c",x"07c84328",x"07c80310
        ",x"074712ec",x"224b328c",x"3fcff3fc",x"3e0f83e0",x"2248a230",x"1645a174",x"40000000",x"
        40000000",
141 x"40000000",x"40000000",x"40000000",x"2b4ae2bc",x"34cd334c",x"3fcff3fc",x"2f8bd350",x"1409d388",x"
        0fc9e380",x"0fc9e380",x"110a1384",x"108a0384",x"0fc9e380",x"0fc9e380",x"0e09b37c",x"0c497378",
        x"0bc96378",x"09c92370",x"0888e364",x"08489348",x"08086334",x"0808532c",x"07c80310",x"078742fc
        ",x"218ae2c4",x"3fcfb3fc",x"3fcff3fc",x"284a2290",x"18061190",x"40000000",x"40000000",x"
        40000000",
142 x"40000000",x"40000000",x"40000000",x"40000000",x"2acac2b4",x"354d6358",x"3fcff3fc",x"348cc368",x"
        1a0a2380",x"0fc9e380",x"0d89a37c",x"0e09b37c",x"0e09b37c",x"0d89a37c",x"0c497378",x"0b094374",
        x"09c92370",x"0888f36c",x"08489348",x"08086334",x"07c83320",x"0787d2fc",x"08c79340",x"2a8c6304
        ",x"3fcfe3fc",x"3fcff3fc",x"2b0ad2b8",x"194671a4",x"40000000",x"40000000",x"40000000",x"
        40000000",
143 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"294a629c",x"330cc334",x"3fcff3fc",x"
        3d0e93bc",x"280b8378",x"16ca3390",x"0d89a37c",x"0a894374",x"08c90370",x"0888f36c",x"0848b350",
        x"08489348",x"0808532c",x"07c82318",x"07c82318",x"08480354",x"1e4a5330",x"38ce7394",x"3fcff3fc
        ",x"3ecfb3ec",x"288a3294",x"19c681ac",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",
144 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"2749d27c",x"2d8b72e0",x"
        3b4ee3b8",x"3fcff3fc",x"3d0e93bc",x"2e8c9384",x"218b1388",x"18ca5390",x"1249f388",x"0f49c380",
        x"0ec98380",x"12c9737c",x"1bca3370",x"294c1358",x"39ce93a0",x"3fcff3fc",x"3fcff3fc",x"36cdc370
        ",x"22c8c238",x"18863198",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",
145 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"24893254",x"
        27c9f284",x"2fcbf300",x"3a8ea3a8",x"3fcff3fc",x"3fcff3fc",x"3fcf83f4",x"3e4f03d4",x"3d4ee3c8",
        x"3e0f13d4",x"3f8f93f4",x"3fcff3fc",x"3fcff3fc",x"3f8fe3f8",x"370dc374",x"288a2290",x"1cc741d8
```

```
",x"1685c178",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
146 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",x"2288a230",x"25896260",x"2a8ab2b0",x"304c230c",x"354d5358",x"384e1384",x"39ce639c",
    x"398e5398",x"374dd378",x"334cd338",x"2ccb42d4",x"25094258",x"1d4761e4",x"17860188",x"40000000
    ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
147 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",x"40000000",x"40000000",x"1c4731d4",x"1e87b1f8",x"20883214",x"21486220",x"20883214",
    x"1fc80208",x"1f07d1fc",x"1c8731d4",x"18c6419c",x"1384f148",x"40000000",x"40000000",x"40000000
    ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
148 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",x"40000000",x"40000000",x"358d535c",x"358d635c",x"358d735c",x"358d635c",x"354d5358",
    x"348d234c",x"334cd338",x"318c6320",x"2f8be300",x"40000000",x"40000000",x"40000000",x"40000000
    ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
149 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",x"378de378",x"37ce0380",x"394e5398",x"3bcee3bc",x"3dcf73dc",x"3ecfb3ec",x"3f0fc3f0",
    x"3ecfb3ec",x"3dcf83e0",x"3bcef3bc",x"380e1384",x"33ccf340",x"2fcbf300",x"2ccb42d4",x"40000000
    ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
150 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"384e1388",x"
    38ce3390",x"3b4ee3b8",x"3f8fe3f8",x"3fcff3fc",x"3c4ef3cc",x"360d5374",x"314c2324",x"2ecb92fc",
    x"2f8bb304",x"330ca340",x"38ce0398",x"3e0f63e8",x"3fcff3fc",x"3c8f23c8",x"340d1344",x"2dcb72e4
    ",x"2a4a92b0",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
151 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"38ce338c",x"3a8ea3a8",x"
    3f4fe3f8",x"3f0fb3f4",x"328c733c",x"1bc6e2a0",x"0c833210",x"0681c1b4",x"044121b8",x"03c0f1b4",
    x"044121b0",x"0601b1a4",x"0ac301bc",x"15c5a220",x"2a0a82a8",x"38ce039c",x"3fcff3fc",x"3b4ed3b8
    ",x"2fcbf300",x"2a0a82a8",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
152 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3c0f03c0",x"3fcff3fc",x"
    354d3364",x"14c53274",x"0541719c",x"03c0f1b0",x"05415258",x"0681a2ec",x"0741d340",x"0781e35c",
    x"0741d340",x"0681a2f4",x"0581627c",x"044111e8",x"04c1717c",x"0f0441a8",x"298a629c",x"3c0ee3cc
    ",x"3fcff3fc",x"320c8324",x"294a529c",x"40000000",x"40000000",x"40000000",x"40000000",x"
    40000000",
153 x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3ccf33cc",x"3fcff3fc",x"298a32e0",x"
    084231b4",x"03c0f1b0",x"0681a2fc",x"10c43390",x"1605839c",x"14c53398",x"12048394",x"10441390",
    x"0f03c38c",x"0e03838c",x"0dc37388",x"0b02c384",x"0701c32c",x"04812214",x"0582013c",x"1c0761e8
    ",x"36cd8380",x"3fcff3fc",x"324ca328",x"27ca0284",x"40000000",x"40000000",x"40000000",x"
    40000000",
154 x"40000000",x"40000000",x"40000000",x"38ce4390",x"3c4f13c4",x"3fcff3fc",x"2408e2b4",x"05417184",x"
    05014254",x"11445390",x"17c5f3a0",x"13c4f398",x"1344d398",x"1284a394",x"12048394",x"10c43390",
    x"0e83a38c",x"0dc37388",x"0c030384",x"0a429380",x"09425380",x"0a429380",x"0681a2e4",x"0401715c
    ",x"1606318c",x"358d3368",x"3fcff3fc",x"2fcc0304",x"25897264",x"40000000",x"40000000",x"
    40000000",
155 x"40000000",x"40000000",x"388e238c",x"3acea3ac",x"3fcff3fc",x"290a22c8",x"05818180",x"060182bc",x"
    17c5f3a0",x"1505439c",x"1344d398",x"13c4f398",x"13c4f398",x"1344d398",x"1284a394",x"12048394",
    x"0fc3f390",x"0e83a38c",x"0cc33388",x"0b02c384",x"09425380",x"0802037c",x"0802037c",x"0741d340
    ",x"0401a170",x"1906f17c",x"388df398",x"3ecfb3ec",x"2b0ad2b8",x"23890244",x"40000000",x"
    40000000",
156 x"40000000",x"40000000",x"394e5394",x"3fcff3fc",x"338cc34c",x"090251a4",x"060182ac",x"190643a4",x"
    1505439c",x"14c53398",x"14c53398",x"13c4f398",x"1344d398",x"13c4f398",x"1344d398",x"1284a394",
    x"11846394",x"0fc3f390",x"0e03838c",x"0c431388",x"0a82a384",x"08c23380",x"0781e36c",x"0781e364
    ",x"0741d338",x"04c26134",x"22c90200",x"3e4f73f0",x"37ce0380",x"25496260",x"40000000",x"
    40000000",
157 x"40000000",x"37cde380",x"3bcf03c0",x"3e4f83e8",x"15853230",x"04c13238",x"17c5f3a0",x"1685a39c",x"
```

```
1505439c",x"1505439c",x"14c53398",x"14c53398",x"13c4f398",x"1344d398",x"13c4f398",x"1344d398",
x"12048394",x"10c43390",x"0f03c38c",x"0cc33388",x"0b02c384",x"09425380",x"07c1f378",x"0781e358
",x"0741d350",x"0681a2ec",x"09841100",x"30cc1308",x"3fcff3fc",x"2ccb42d4",x"20883218",x"
40000000",
158 x"40000000",x"384e2388",x"3fcff3fc",x"304bd310",x"0681e1a8",x"0f03c38c",x"188623a4",x"1605839c",x"
1585639c",x"1585639c",x"1505439c",x"14c53398",x"14c53398",x"13c4f398",x"13c4f398",x"13c4f398",
x"1284a394",x"12048394",x"0fc3f390",x"0e03838c",x"0c431388",x"0a429380",x"0802037c",x"0781e36c
",x"0741d348",x"0741d348",x"054261f4",x"1cc82190",x"3e4f63f0",x"370dc374",x"2348e240",x"
40000000",
159 x"360d8368",x"39ce83a0",x"3fcff3fc",x"1b06724c",x"064192c8",x"1a4693a8",x"1685a39c",x"1685a39c",x"
1685a39c",x"1685a39c",x"1585639c",x"1505439c",x"14c53398",x"14c53398",x"1344d398",x"13c4f398",
x"1344d398",x"12048394",x"10c43390",x"0e83a38c",x"0c431388",x"0b02c384",x"08c23380",x"0781e36c
",x"0741d350",x"0701c334",x"06c1b300",x"0bc5411c",x"350d0344",x"3ecfb3ec",x"280a1288",x"
40000000",
160 x"364d9368",x"3c8f23c8",x"3a8e73b0",x"0d43b1f4",x"0f03c38c",x"190643a4",x"1745d3a0",x"1745d3a0",x"
16c5b3a0",x"1685a39c",x"1685a39c",x"1585639c",x"1505439c",x"14c53398",x"13c4f398",x"13c4f398",
x"1344d398",x"1284a394",x"10c43390",x"0f03c38c",x"0cc33388",x"0b02c384",x"09425380",x"0781e374
",x"0741d350",x"0701c334",x"0701c32c",x"0683f194",x"28caa26c",x"3fcff3fc",x"2d8b72e0",x"1
cc741dc",
161 x"364da368",x"3e8fa3e8",x"340c7338",x"0882d25c",x"184613a0",x"17c5f3a0",x"17c5f3a0",x"17c5f3a0",x"
1745d3a0",x"16c5b3a0",x"1685a39c",x"1585639c",x"1505439c",x"14c53398",x"13c4f398",x"1344d398",
x"1344d398",x"1304c394",x"11846394",x"0f83e38c",x"0dc37388",x"0b82e384",x"09425380",x"07c1f378
",x"0781e358",x"0741d338",x"0701c31c",x"0602f224",x"1ec941d0",x"3fcff3fc",x"328ca32c",x"1
e4791f0",
162 x"364d9368",x"3f8fe3f8",x"2ecb02d4",x"07c27314",x"1a0683a4",x"184613a0",x"188623a4",x"184613a0",x"
17c5f3a0",x"1745d3a0",x"1685a39c",x"1605839c",x"1585639c",x"14c53398",x"14451398",x"1344d398",
x"13c4f398",x"1344d398",x"12048394",x"0fc3f390",x"0e03838c",x"0c030384",x"09425380",x"07c1f378
",x"0781e364",x"0741d338",x"0701c31c",x"06422298",x"1848d184",x"3f8f73f4",x"360d8364",x"1
f87f204",
163 x"35cd7360",x"3fcff3fc",x"2cca92b4",x"0b02c384",x"1a0683a4",x"188623a4",x"190643a4",x"188623a4",x"
17c5f3a0",x"1745d3a0",x"1685a39c",x"1685a39c",x"1585639c",x"14c53398",x"14c53398",x"1344d398",
x"13c4f398",x"1344d398",x"12048394",x"0fc3f390",x"0e03838c",x"0c030384",x"09c27380",x"0802037c
",x"0781e364",x"0741d338",x"0701c31c",x"064192d0",x"1588e178",x"3e4f33dc",x"378de378",x"
2008120c",
164 x"350d5354",x"3f8fe3f8",x"2d4ae2c8",x"0fc41388",x"190643a4",x"188623a4",x"188623a4",x"188623a4",x"
17c5f3a0",x"1745d3a0",x"1685a39c",x"1605839c",x"1585639c",x"14c53398",x"14451398",x"1344d398",
x"13c4f398",x"1344d398",x"12048394",x"0fc3f390",x"0e03838c",x"0c030384",x"09425380",x"07c1f378
",x"0781e364",x"0741d338",x"0701c31c",x"0641b2c0",x"16094190",x"3ecf53e8",x"374dd378",x"1
f87f204",
165 x"344d1348",x"3ecfb3ec",x"304b72f4",x"0f849368",x"188623a4",x"17c5f3a0",x"17c5f3a0",x"17c5f3a0",x"
1745d3a0",x"16c5b3a0",x"1685a39c",x"1585639c",x"1505439c",x"14c53398",x"13c4f398",x"1344d398",
x"1344d398",x"1304c394",x"11846394",x"0f83e38c",x"0dc37388",x"0b82e384",x"09425380",x"07c1f378
",x"0781e358",x"0741d338",x"06c1b318",x"0602128c",x"1a09e1d0",x"3fcfd3fc",x"354d5358",x"1
d8771e4",
166 x"32ccb330",x"3d0f43d4",x"360cc348",x"0f050324",x"184613a0",x"1745d3a0",x"1745d3a0",x"1745d3a0",x"
16c5b3a0",x"1685a39c",x"1685a39c",x"1585639c",x"1505439c",x"14c53398",x"13c4f398",x"13c4f398",
x"1344d398",x"1284a394",x"10c43390",x"0f03c38c",x"0d435388",x"0b02c384",x"09425380",x"0781e374
",x"0781e358",x"0741d338",x"06c1b318",x"0602f230",x"218ae23c",x"3fcff3fc",x"310c5314",x"1
a86b1b8",
167 x"30cc4314",x"39ce73a0",x"3c4e73b0",x"140662dc",x"16c5b3a0",x"1685a39c",x"16c5b3a0",x"1685a39c",x"
1685a39c",x"1685a39c",x"1585639c",x"1505439c",x"14c53398",x"14c53398",x"13c4f398",x"13c4f398",
x"1344d398",x"12048394",x"10c43390",x"0e83a38c",x"0c431388",x"0b02c384",x"08c23380",x"0781e36c
",x"0741d350",x"0701c334",x"06c1b310",x"06c461d8",x"2d8ca2e8",x"3fcff3fc",x"2acac2b4",x"1785
f188",
168 x"2d8b62dc",x"358d6358",x"3fcfd3f8",x"230902c8",x"14451398",x"1605839c",x"1685a39c",x"1685a39c",x"
1605839c",x"1585639c",x"1505439c",x"14c53398",x"14c53398",x"13c4f398",x"1344d398",x"13c4f398",
x"1304c394",x"12048394",x"0fc3f390",x"0e03838c",x"0c431388",x"0a82a384",x"08421380",x"0781e36c
",x"0741d350",x"0701c32c",x"0681a2e4",x"0d4701b4",x"3aceb3ac",x"3ecfa3ec",x"2348e240",x"0
```

```
      cc360f0",
169 x"40000000",x"314c5318",x"3dcf73dc",x"354c9340",x"10c5333c",x"1685a39c",x"1585639c",x"1585639c",x"
      1505439c",x"1505439c",x"14c53398",x"14c53398",x"13c4f398",x"1344d398",x"13c4f398",x"1344d398",
      x"12048394",x"10c43390",x"0f03c38c",x"0cc33388",x"0b02c384",x"09425380",x"07c1f378",x"0781e364
      ",x"0741d338",x"0701c324",x"05c2b244",x"224b224c",x"3fcff3fc",x"34cd3350",x"1c4711d0",x"
      40000000",
170 x"40000000",x"2e0b82e8",x"364da368",x"3f4f93ec",x"20c8a2e4",x"13c4f398",x"1585639c",x"14c53398",x"
      14c53398",x"14c53398",x"14c53398",x"13c4f398",x"1344d398",x"13c4f398",x"1344d398",x"1284a394",
      x"12048394",x"0fc3f390",x"0e03838c",x"0c431388",x"0a82a384",x"08c23380",x"0781e374",x"0741d350
      ",x"0741d338",x"0681a2e4",x"0b8631e4",x"398e839c",x"3fcff3fc",x"2749e280",x"1685c178",x"
      40000000",
171 x"40000000",x"40000000",x"2f4be2fc",x"3ccf43d0",x"390d8378",x"15867320",x"14451398",x"14451398",x"
      14451398",x"13c4f398",x"13c4f398",x"1344d398",x"13c4f398",x"1344d398",x"1304c394",x"12048394",
      x"10441390",x"0e83a38c",x"0cc33388",x"0b02c384",x"09425380",x"0802037c",x"0781e36c",x"0741d348
      ",x"06c1b318",x"06439218",x"2c0c92dc",x"3fcff3fc",x"350d4350",x"1b8701c8",x"40000000",x"
      40000000",
172 x"40000000",x"40000000",x"2b8af2c0",x"320c8324",x"3fcff3fc",x"328c0334",x"13c5b35c",x"1304c394",x"
      1344d398",x"13c4f398",x"13c4f398",x"13c4f398",x"1344d398",x"1284a394",x"12048394",x"10c43390",
      x"0f03c38c",x"0dc37388",x"0c030384",x"0a429380",x"08421380",x"0781e36c",x"0741d350",x"0701c324
      ",x"06028268",x"224b2274",x"3fcff3fc",x"3e0f83e0",x"2248a230",x"1645a174",x"40000000",x"
      40000000",
173 x"40000000",x"40000000",x"40000000",x"2b4ae2bc",x"34cd334c",x"3fcff3fc",x"308b9338",x"14c5e364",x"
      12048394",x"1284a394",x"1344d398",x"1304c394",x"1284a394",x"12048394",x"10c43390",x"0f03c38c",
      x"0e03838c",x"0c431388",x"0a82a384",x"09425380",x"07c1f378",x"0781e364",x"0701c31c",x"06428290
      ",x"214aa294",x"3fcfb3fc",x"3fcff3fc",x"284a2290",x"18061190",x"40000000",x"40000000",x"
      40000000",
174 x"40000000",x"40000000",x"40000000",x"40000000",x"2acac2b4",x"354d6358",x"3fcff3fc",x"350ca35c",x"
      1a878348",x"12048394",x"10441390",x"10c43390",x"10c43390",x"10441390",x"0e83a38c",x"0dc37388",
      x"0c431388",x"0b02c384",x"09425380",x"0802037c",x"0741d348",x"0681a2f4",x"0903e2c4",x"2a8c42f0
      ",x"3fcfe3fc",x"3fcff3fc",x"2b0ad2b8",x"194671a4",x"40000000",x"40000000",x"40000000",x"
      40000000",
175 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"294a629c",x"330cc334",x"3fcff3fc",x"
      3d0e93bc",x"294aa348",x"1806a370",x"10441390",x"0d435388",x"0b82e384",x"0b02c384",x"09c27380",
      x"08c23380",x"0781e36c",x"0741d338",x"0701c32c",x"0882e334",x"1dc952dc",x"38ce7394",x"3fcff3fc
      ",x"3ecfb3ec",x"288a3294",x"19c681ac",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
176 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"2749d27c",x"2d8b72e0",x"
      3b4ee3b8",x"3fcff3fc",x"3d0e93bc",x"2f4c1364",x"21c9634c",x"19070364",x"13c54384",x"11847390",
      x"11048378",x"1345b348",x"1b884320",x"294b9328",x"39ce93a0",x"3fcff3fc",x"3fcff3fc",x"36cdc370
      ",x"22c8c238",x"18863198",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
177 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"24893254",x"
      27c9f284",x"2fcbf300",x"3a8ea3a8",x"3fcff3fc",x"3fcff3fc",x"3fcf83f4",x"3e4f03d4",x"3d4ee3c8",
      x"3e0f13d4",x"3f8f93f4",x"3fcff3fc",x"3fcff3fc",x"3f8fe3f8",x"370dc374",x"288a2290",x"1cc741d8
      ",x"1685c178",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
178 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"2288a230",x"25896260",x"2a8ab2b0",x"304c230c",x"354d5358",x"384e1384",x"39ce639c",
      x"398e5398",x"374dd378",x"334cd338",x"2ccb42d4",x"25094258",x"1d4761e4",x"17860188",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
179 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"40000000",x"40000000",x"1c4731d4",x"1e87b1f8",x"20883214",x"21486220",x"20883214",
      x"1fc80208",x"1f07d1fc",x"1c8731d4",x"18c6419c",x"1384f148",x"40000000",x"40000000",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
180 x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"40000000",x"40000000",x"358d535c",x"358d635c",x"358d735c",x"358d635c",x"354d5358",
```

```
        x"348d234c",x"334cd338",x"318c6320",x"2f8be300",x"40000000",x"40000000",x"40000000",x"40000000
        ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",
181     x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",x"378de378",x"37ce0380",x"394e5398",x"3bcee3bc",x"3dcf73dc",x"3ecfb3ec",x"3f0fc3f0",
        x"3ecfb3ec",x"3dcf83e0",x"3bcef3bc",x"380e1384",x"33ccf340",x"2fcbf300",x"2ccb42d4",x"40000000
        ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",
182     x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"384e1388",x"
        38ce3390",x"3b4ee3b8",x"3f8fe3f8",x"3fcff3fc",x"3c4ef3cc",x"360d5374",x"314c2324",x"2ecb92fc",
        x"2f8bb304",x"330ca340",x"38ce0398",x"3e0f63e8",x"3fcff3fc",x"3c8f23c8",x"340d1344",x"2dcb72e4
        ",x"2a4a92b0",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",
183     x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"38ce338c",x"3a8ea3a8",x"
        3f4fe3f8",x"3f0fb3f4",x"328c733c",x"1cc8929c",x"11897284",x"0d8a929c",x"0b0b52b8",x"0a4b82c4",
        x"0b0b52b8",x"0d8a7298",x"10095278",x"1747f268",x"2a0a82a8",x"38ce039c",x"3fcff3fc",x"3b4ed3b8
        ",x"2fcbf300",x"2a0a82a8",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",
184     x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3c0f03c0",x"3fcff3fc",x"
        354d3364",x"17487288",x"0c8ac2a0",x"0a0b92c8",x"0a0bd2d4",x"0a4c22e8",x"0a8c32f0",x"0a8c52f4",
        x"0a8c32f0",x"0a4c22e8",x"0a0bd2d4",x"0a0b92c8",x"0c8ac2a0",x"12080248",x"298a629c",x"3c0ee3cc
        ",x"3fcff3fc",x"320c8324",x"294a529c",x"40000000",x"40000000",x"40000000",x"40000000",x"
        40000000",
185     x"40000000",x"40000000",x"40000000",x"40000000",x"390e4394",x"3ccf33cc",x"3fcff3fc",x"298a12d8",x"
        0ec9e288",x"0a0b92c8",x"0a4c22e8",x"0b0cb310",x"0b8d1324",x"0b4d0320",x"0b0cf31c",x"0b0cb310",
        x"0b0cb310",x"0b0cb310",x"0acc8300",x"0a8c32f0",x"0a0bd2d4",x"0dc99268",x"1c07d214
        ",x"36cd8380",x"3fcff3fc",x"324ca328",x"27ca0284",x"40000000",x"40000000",x"40000000",x"
        40000000",
186     x"40000000",x"40000000",x"40000000",x"38ce4390",x"3c4f13c4",x"3fcff3fc",x"240932b4",x"0ccaa29c",x"
        0a0bd2d4",x"0b0cb310",x"0bcd2328",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",
        x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0acc8300",x"0a4c22e8",x"0c8aa29c
        ",x"164721ec",x"358d3368",x"3fcff3fc",x"2fcc0304",x"25897264",x"40000000",x"40000000",x"
        40000000",
187     x"40000000",x"40000000",x"388e238c",x"3acea3ac",x"3fcff3fc",x"290a02c8",x"0d0a9298",x"0a4c22e8",x"
        0bcd2328",x"0b8d1324",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
        x"0b0cb310",x"0b0cb310",x"0acca308",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4",x"0a8c32f0
        ",x"0cca9298",x"190741b0",x"388df398",x"3ecfb3ec",x"2b0ad2b8",x"23890244",x"40000000",x"
        40000000",
188     x"40000000",x"40000000",x"394e5394",x"3fcff3fc",x"338cc34c",x"0ec97274",x"0a4c02e4",x"0ccd232c",x"
        0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
        x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4
        ",x"0a8c32f0",x"0e496264",x"22c90200",x"3e4f73f0",x"37ce0380",x"25496260",x"40000000",x"
        40000000",
189     x"40000000",x"37cde380",x"3bcf03c0",x"3e4f83e8",x"1647e26c",x"0a0bd2d4",x"0bcd2328",x"0b8d1324",x"
        0b8d1324",x"0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
        x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acca308",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4
        ",x"0a8c52f4",x"0a4c22e8",x"0e4731f0",x"30cc1308",x"3fcff3fc",x"2ccb42d4",x"20883218",x"
        40000000",
190     x"40000000",x"384e2388",x"3fcff3fc",x"304bd310",x"0dca428c",x"0b0cb310",x"0ccd232c",x"0b8d1324",x"
        0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
        x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c52f4",x"0a8c52f4
        ",x"0a8c32f0",x"0a8c32f0",x"0e8aa2a8",x"1cc82190",x"3e4f63f0",x"370dc374",x"2348e240",x"
        40000000",
191     x"360d8368",x"39ce83a0",x"3fcff3fc",x"1ac7e270",x"0a4c22e8",x"0d8d332c",x"0b8d1324",x"0b8d1324",x"
        0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
        x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4
        ",x"0a8c52f4",x"0a8c32f0",x"0a4c22e8",x"0f8701d4",x"350d0344",x"3ecfb3ec",x"280a1288",x"
        40000000",
```

```
192 x"364d9368",x"3c8f23c8",x"3a8e73b0",x"1208d270",x"0b0cb310",x"0ccd232c",x"0bcd2328",x"0bcd2328",x"
      0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acca308",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a8c32f0",x"1108d264",x"28caa26c",x"3fcff3fc",x"2d8b72e0",x"1
      cc741dc",
193 x"364da368",x"3e8fa3e8",x"340c7338",x"100aa2b0",x"0bcd2328",x"0bcd2328",x"0bcd2328",x"0bcd2328",x"
      0bcd2328",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a8c32f0",x"0f8a92ac",x"1ec941d0",x"3fcff3fc",x"328ca32c",x"1
      e4791f0",
194 x"364d9368",x"3f8fe3f8",x"2ecb02d4",x"0d8bc2e0",x"0d8d332c",x"0bcd2328",x"0ccd232c",x"0bcd2328",x"
      0bcd2328",x"0bcd2328",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a8c32f0",x"0d0b92d0",x"1848d184",x"3f8f73f4",x"360d8364",x"1
      f87f204",
195 x"35cd7360",x"3fcff3fc",x"2cca92b4",x"0acc8300",x"0d8d332c",x"0ccd232c",x"0ccd232c",x"0ccd232c",x"
      0bcd2328",x"0bcd2328",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a8c32f0",x"0a4c22e8",x"1588e178",x"3e4f33dc",x"378de378",x"
      2008120c",
196 x"350d5354",x"3f8fe3f8",x"2d4ae2c8",x"0b8cc314",x"0ccd232c",x"0ccd232c",x"0ccd232c",x"0ccd232c",x"
      0bcd2328",x"0bcd2328",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a8c32f0",x"0acbf2e0",x"16094190",x"3ecf53e8",x"374dd378",x"1
      f87f204",
197 x"344d1348",x"3ecfb3ec",x"304b72f4",x"0ecc32f8",x"0ccd232c",x"0bcd2328",x"0bcd2328",x"0bcd2328",x"
      0bcd2328",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a4c22e8",x"0d0b92d0",x"1a09e1d0",x"3fcfd3fc",x"354d5358",x"1
      d8771e4",
198 x"32ccb330",x"3d0f43d4",x"360cc348",x"148b02cc",x"0bcd2328",x"0bcd2328",x"0bcd2328",x"0bcd2328",x"
      0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a4c22e8",x"0f8a92ac",x"218ae23c",x"3fcff3fc",x"310c5314",x"1
      a86b1b8",
199 x"30cc4314",x"39ce73a0",x"3c4e73b0",x"1a49a2b0",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"
      0b8d1324",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a4c22e8",x"1248f274",x"2d8ca2e8",x"3fcff3fc",x"2acac2b4",x"1785
      f188",
200 x"2d8b62dc",x"358d6358",x"3fcfd3f8",x"240962bc",x"0b0cf31c",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"
      0b8d1324",x"0b8d1324",x"0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4
      ",x"0a8c52f4",x"0a8c32f0",x"0a4c22e8",x"1448122c",x"3aceb3ac",x"3ecfa3ec",x"2348e240",x"0
      cc360f0",
201 x"40000000",x"314c5318",x"3dcf73dc",x"354c9340",x"140b42d8",x"0b8d1324",x"0b8d1324",x"0b8d1324",x"
      0b8d1324",x"0b8d1324",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acca308",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4
      ",x"0a8c32f0",x"0a8c32f0",x"0f0ac2b0",x"224b224c",x"3fcff3fc",x"34cd3350",x"1c4711d0",x"
      40000000",
202 x"40000000",x"2e0b82e8",x"364da368",x"3f4f93ec",x"224982c0",x"0b0cf31c",x"0b8d1324",x"0b4d0320",x"
      0b4d0320",x"0b4d0320",x"0b4d0320",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
      x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4
      ",x"0a8c32f0",x"0a4c22e8",x"14884258",x"398e839c",x"3fcff3fc",x"2749e280",x"1685c178",x"
      40000000",
203 x"40000000",x"40000000",x"2f4be2fc",x"3ccf43d0",x"390d8378",x"198a92c8",x"0b0cf31c",x"0b0cf31c",x"
      0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",
      x"0b0cb310",x"0b0cb310",x"0acca308",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4",x"0a8c32f0
```

```
      ",x"0a4c22e8",x"1149f298",x"2c0c92dc",x"3fcff3fc",x"350d4350",x"1b8701c8",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"2b8af2c0",x"320c8324",x"3fcff3fc",x"320bd32c",x"130bc2e8",x"0b0cf31c",x"
      0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",
      x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4",x"0a8c32f0
      ",x"0e4b22c0",x"23cae284",x"3fcff3fc",x"3e0f83e0",x"2248a230",x"1645a174",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"40000000",x"2b4ae2bc",x"34cd334c",x"3fcff3fc",x"2f8b431c",x"12cbe2f0",x"
      0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cf31c",x"0b0cb310",x"0b0cb310",
      x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c52f4",x"0a8c32f0",x"0e4b42c8
      ",x"248a52a8",x"3fcfb3fc",x"3fcff3fc",x"284a2290",x"18061190",x"40000000",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"40000000",x"40000000",x"2acac2b4",x"354d6358",x"3fcff3fc",x"344c5348",x"
      1a0b02d8",x"0b0cf31c",x"0b0cb310",x"0b0cb310",x"0b0cb310",x"0b0cb310",x"0b0cb310",x"0b0cb310",
      x"0acc8300",x"0acc8300",x"0a8c72fc",x"0a8c52f4",x"0a8c32f0",x"0a4c22e8",x"12ca82b8",x"2bcbe2f0
      ",x"3fcfe3fc",x"3fcff3fc",x"2b0ad2b8",x"194671a4",x"40000000",x"40000000",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"294a629c",x"330cc334",x"3fcff3fc",x"
      3d0e93bc",x"28cad2f8",x"134c12f8",x"0b0cb310",x"0b0cb310",x"0acc8300",x"0acc8300",x"0a8c72fc",
      x"0a8c72fc",x"0a8c52f4",x"0a8c32f0",x"0a8c32f0",x"0ecbb2e0",x"2289f2c4",x"38ce7394",x"3fcff3fc
      ",x"3ecfb3ec",x"288a3294",x"19c681ac",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"2749d27c",x"2d8b72e0",x"
      3b4ee3b8",x"3fcff3fc",x"3d0e93bc",x"2e4b9320",x"21cb02e8",x"164bc2f0",x"0dcca310",x"0b4ce318",
      x"0d4c9308",x"140ba2e4",x"1e8a92d0",x"2b0b2300",x"39ce93a0",x"3fcff3fc",x"3fcff3fc",x"36cdc370
      ",x"22c8c238",x"18863198",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"24893254",x"
      27c9f284",x"2fcbf300",x"3a8ea3a8",x"3fcff3fc",x"3fcff3fc",x"3fcf83f4",x"3e4f03d4",x"3d4ee3c8",
      x"3e0f13d4",x"3f8f93f4",x"3fcff3fc",x"3fcff3fc",x"3f8fe3f8",x"370dc374",x"288a2290",x"1cc741d8
      ",x"1685c178",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"2288a230",x"25896260",x"2a8ab2b0",x"304c230c",x"354d5358",x"384e1384",x"39ce639c",
      x"398e5398",x"374dd378",x"334cd338",x"2ccb42d4",x"25094258",x"1d4761e4",x"17860188",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",
x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",x"40000000",x"40000000",x"1c4731d4",x"1e87b1f8",x"20883214",x"21486220",x"20883214",
      x"1fc80208",x"1f07d1fc",x"1c8731d4",x"18c6419c",x"1384f148",x"40000000",x"40000000",x"40000000
      ",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"40000000",x"
      40000000",others=>x"00000000");


begin

process (Clk)
begin

index<=addrx*32+addry+(8192-(printspritenumber-1)*1024);


if rising_edge(Clk) then
if en = '1' then
data <= ROM1(index);
end if;
end if;
```

```
end process;


end imp;
```

Listing 13: FPGA/rombut1.vhd

```
-- This file is part of The Awesome Guitar Game.
--
--    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
--    it under the terms of the GNU General Public License as published by
--    the Free Software Foundation, either version 3 of the License, or
--    (at your option) any later version.
--
--    The Awesome Guitar Game is distributed in the hope that it will be useful,
--    but WITHOUT ANY WARRANTY; without even the implied warranty of
--    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
--    GNU General Public License for more details.
--
--    You should have received a copy of the GNU General Public License
--    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
--
--
-----------------------------------------------------------------------
-- The awesome guitar game (A clone of Guitar Hero for FPGA #
-----------------------------------------------------------------------
-- EMBEDDED SYSTEM PROJECT
-- Columbia Unviersity Spring 2012
--
-- Avijit Singh Wasu -- asw2156@columbia.edu
-- Laurent Charignon -- lc2817@columbia.edu
--
-- Licensed under the GPL license
-- Have a look at the license file in the root of
-- the project to have more details about the license
--
-----------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity rombut is

  port (
    clk , en : in std_logic;
    data : out unsigned (24 downto 0);
    addr: in unsigned (9 downto 0)
    );
end rombut;



architecture ar of rombut is


type rom_type is array (1023 downto 0) of unsigned(27 downto 0);
```

```
50 constant ROM : rom_type :=(x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"
       1000000",x"1000000",x"1eae7e8",x"1ffffff",x"1000000",x"1000000",x"1f0d9d6",x"0dadce5",x"0
       cbd0d5",x"0c7cacd",x"0c7cbd0",x"0cfd1d3",x"0d0d0cd",x"1574a26",x"1000000",x"1000000",x"1ffffff
       ",x"13e444c",x"1000000",x"1000000",x"11a1c23",x"1000000",x"1000000",x"1000000",x"1000000",x"
       1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1f8f9f9",x"
       1000000",x"1000000",x"1000000",x"0f8ffff",x"0eeffff",x"0e2f7fa",x"0dcedf1",x"0d7e6e9",x"0
       c9d7de",x"0afbdc7",x"095a2af",x"08e99a5",x"08e99a3",x"08d99a1",x"0758288",x"1000000",x"1000000
       ",x"1000000",x"1ffffff",x"1000000",x"11a1c23",x"1000000",x"1000000",x"1000000",x"1000000",x"
       1000000",x"1000000",x"1000000",x"1eaf1f2",x"1000000",x"1fdffff",x"1000000",x"1000000",x"0
       ffffff",x"0f9f8f8",x"0e9e6e5",x"0dccbc8",x"0d4afaf",x"0cfa3a4",x"0c9aca9",x"0cbc3c1",x"0dadada
       ",x"0e5e7e7",x"0c9d1d8",x"08894a1",x"0778592",x"0838e9b",x"0768090",x"04e5b67",x"1000000",x"
       1000000",x"132393a",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"
       1000000",x"1000000",x"1d5fef9",x"1000000",x"1000000",x"0ffffff",x"0edebe9",x"0f4a4a8",x"0
       ee747a",x"0ff2b30",x"0ff3025",x"0ff2923",x"0ff211b",x"0ff1d11",x"0fe1800",x"0f20000",x"0c3595c
       ",x"0c3bbb9",x"0f5f5f5",x"0c3cbd0",x"077828f",x"0737e8b",x"0697783",x"04f5e66",x"0000000",x"
       1000000",x"1000000",x"115171f",x"11a1c23",x"1000000",x"1000000",x"1000000",x"1000000",x"
       1000000",x"1fe0635",x"1000000",x"0f3f1f1",x"0f19b9b",x"0f9575f",x"0ff3e39",x"0ff3833",x"0
       ff2d37",x"0ff2732",x"0ff212c",x"0ff1b26",x"0ff131f",x"0ff0f1b",x"0ff070d",x"0fa0c00",x"0ea2100
       ",x"0bd3f42",x"0d4d1d0",x"0e1e2e3",x"0818c97",x"0747d88",x"0666f78",x"04f555c",x"0121317",x"
       1000000",x"1000000",x"11a1c23",x"1000000",x"1000000",x"1000000",x"1ee3858",x"1000000",x"
       1000000",x"0dcd8d8",x"0fb777a",x"0ff4343",x"0ff3d46",x"0ff3741",x"0ff2f39",x"0ff2732",x"0
       ff1f2a",x"0ff1b26",x"0ff1722",x"0ff131f",x"0ff0d19",x"0ff0713",x"0fe000c",x"0fa000c",x"0f20000
       ",x"0d60f00",x"0acaaac",x"0cfd0d1",x"0848e9a",x"0707982",x"05e656d",x"042454c",x"0000000",x"
       1000000",x"1000000",x"11a1c23",x"1000000",x"1ff0000",x"1000000",x"1000000",x"0c0b6b4",x"0
       fd6168",x"0ff4545",x"0ff3f48",x"0ff3741",x"0ff2b35",x"0ff212c",x"0ff1b26",x"0ff131f",x"0ff111d
       ",x"0ff0f1b",x"0ff0d19",x"0ff0b17",x"0ff0713",x"0ff030f",x"0fa000c",x"0f2000b",x"0ec0000",x"0
       d02200",x"0938386",x"0adaeaf",x"0828e99",x"06f767f",x"05a5c62",x"0353338",x"0000000",x"1000000
       ",x"1000000",x"1000000",x"1e3553c",x"1000000",x"1a0ffff",x"0ff5357",x"0ff4750",x"0ff414a",x"0
       ff353f",x"0ff2933",x"0ff1b26",x"0ff131f",x"0ff0f1b",x"0ff0b17",x"0ff0915",x"0ff0915",x"0ff0915
       ",x"0ff0713",x"0ff0713",x"0ff030f",x"0fe000c",x"0f6000c",x"0ec000b",x"0e6000b",x"0ce1800",x"
       0957573",x"0999c9c",x"07e8994",x"075757a",x"0545153",x"0222227",x"1000000",x"1000000",x"1181
       b23",x"1f1093c",x"1000000",x"0f15c55",x"0ff4b54",x"0ff434c",x"0ff3741",x"0ff2933",x"0ff1b26",x
       "0ff111d",x"0ff0b17",x"0ff0915",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0
       ff0713",x"0ff0713",x"0ff030f",x"0fa000c",x"0f0000b",x"0e4000b",x"0de000a",x"0cc0a00",x"095847f
       ",x"0838f94",x"07d8289",x"07a7574",x"03d3b3e",x"003060d",x"1000000",x"1181a20",x"1000000",x"09
       d8a83",x"0ff4b50",x"0ff4750",x"0ff3b45",x"0ff2b35",x"0ff1b26",x"0ff0f1b",x"0ff0b17",x"0ff0713"
       ,x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0
       ff0511",x"0fe000c",x"0f6000c",x"0e6000b",x"0de000a",x"0d20000",x"0b63924",x"0878787",x"07d878f
       ",x"0948b85",x"06b6160",x"01c1c20",x"1000000",x"1000000",x"1000000",x"0fb504f",x"0ff4952",x"0
       ff414a",x"0ff333d",x"0ff1f2a",x"0ff131f",x"0ff0b17",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
       ",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff030f",x"0
       fa000c",x"0ec000b",x"0de000a",x"0d4000a",x"0cc0500",x"09e6e68",x"078818a",x"0958f8a",x"0988a7c
       ",x"02f2d2e",x"0070b11",x"1000000",x"0946a7e",x"0ff494e",x"0ff454e",x"0ff3b45",x"0ff2b35",x"0
       ff1722",x"0ff0b17",x"0ff0915",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
       ",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0511",x"0fe000c",x"0f0000b",x"0
       e2000b",x"0d6000a",x"0ca0005",x"0b2442e",x"07f8080",x"0908b89",x"0b2a08a",x"0534d47",x"00f1014
       ",x"1000000",x"09e7a70",x"0ff4750",x"0ff414a",x"0ff333d",x"0ff232e",x"0ff111d",x"0ff0915",x"0
       ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
       ",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff010d",x"0f2000b",x"0e4000b",x"0d6000a",x"0
       cc000e",x"0c5200f",x"07c7879",x"0828385",x"0b2a290",x"0756c5c",x"0111014",x"0283143",x"0cc675a
       ",x"0ff434c",x"0ff3d46",x"0ff2f39",x"0ff1d28",x"0ff0d19",x"0ff0713",x"0ff0713",x"0ff0713",x"0
       ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
       ",x"0ff0713",x"0ff0713",x"0ff030f",x"0f6000c",x"0e6000b",x"0d8000a",x"0ce000f",x"0ca0a00",x"
       086716c",x"070777b",x"0a3958b",x"086796b",x"018181b",x"01d2125",x"0f44f48",x"0ff414a",x"0
       ff3943",x"0ff2b35",x"0ff1924",x"0ff0b17",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
       ",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0
       ff0713",x"0ff0511",x"0f8000c",x"0e8000b",x"0d8000a",x"0ce000a",x"0ca0500",x"095685f",x"0666b6d
       ",x"08d847f",x"087776c",x"0222022",x"0171b1b",x"0fc4149",x"0ff3f48",x"0ff3741",x"0ff2732",x"0
```

```
ff1722",x"0ff0b17",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0511",x"0
f8000c",x"0e8000b",x"0d8000a",x"0ce000a",x"0ca0500",x"09e5e54",x"05f6264",x"0787270",x"0726763
",x"0252325",x"014161c",x"0f34244",x"0ff3b45",x"0ff333d",x"0ff2732",x"0ff1722",x"0ff0b17",x"0
ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0f8000c",x"0e6000b",x"0
d6000a",x"0ce000f",x"0c80500",x"0956257",x"05a5d5f",x"0676462",x"05d5752",x"0232025",x"015171c
",x"0c96155",x"0ff3943",x"0ff333d",x"0ff2530",x"0ff1722",x"0ff0b17",x"0ff0713",x"0ff0713",x"0
ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff030f",x"0f6000c",x"0e4000b",x"0d4000a",x"0ce000a",x"0
c80500",x"07e685e",x"0555859",x"0585656",x"0484545",x"0201f1f",x"0191b1d",x"0976d65",x"0ff3741
",x"0ff2f39",x"0ff2530",x"0ff1722",x"0ff0b17",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0
ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0713",x"0ff030f",x"0f2000b",x"0e0000b",x"0d2000a",x"0ce000a",x"0ca0a00",x"06a6462",x"
0535456",x"04d4c4c",x"0393636",x"01d1b1a",x"0191d24",x"07d5774",x"0ff333d",x"0ff2f39",x"0
ff2530",x"0ff1924",x"0ff0d19",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0
fe000c",x"0ec000b",x"0da000a",x"0d0000a",x"0ca000a",x"0bd2115",x"05f5d5f",x"04f5152",x"0434142
",x"02d2a2a",x"01a1a19",x"0171c2e",x"0a9133a",x"0fc2f38",x"0ff2b35",x"0ff2530",x"0ff1b26",x"0
ff0f1b",x"0ff0b17",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff030f",x"0f8000c",x"0e6000b",x"0
d6000a",x"0ce000a",x"0ca0000",x"0a14c3b",x"0535152",x"04b4b4b",x"03c3a3a",x"0252324",x"01a1a1e
",x"0161c2b",x"1000000",x"08a666d",x"0ff2732",x"0ff232e",x"0ff1b26",x"0ff131f",x"0ff0b17",x"0
ff0915",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0713",x"0ff0713",x"0ff0511",x"0fe000c",x"0ee000b",x"0dc000a",x"0d0000a",x"0cc000a",x"0
ca0500",x"0615b5c",x"0505253",x"0464445",x"0322f2f",x"01f1d1b",x"0191b20",x"1000000",x"1000000
",x"08e1941",x"0f3282e",x"0ff212c",x"0ff1b26",x"0ff131f",x"0ff0f1b",x"0ff0b17",x"0ff0713",x"0
ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0511
",x"0ff010d",x"0f4000b",x"0e4000b",x"0d4000a",x"0ce000a",x"0ca0500",x"09b432f",x"0544e4f",x"
0505051",x"03f3b3b",x"0292526",x"01d1d18",x"0161a25",x"1000000",x"1c20029",x"1000000",x"06
d4c6a",x"0ff1b26",x"0ff1924",x"0ff131f",x"0ff0f1b",x"0ff0b17",x"0ff0915",x"0ff0713",x"0ff0713"
,x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff030f",x"0fe000c",x"0f6000c",x"0
e6000b",x"0d8000a",x"0ce000a",x"0cc000a",x"0c60500",x"0544f54",x"0585a5c",x"04b494a",x"0333031
",x"0211f1b",x"0191b1d",x"113192f",x"1000000",x"12b3825",x"1000000",x"100ffff",x"0815f6f",x"0
ff1319",x"0ff0f1b",x"0ff0d19",x"0ff0915",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713",x"0ff0713
",x"0ff0511",x"0ff0511",x"0ff030f",x"0fa000c",x"0f2000b",x"0e6000b",x"0da000a",x"0d0000a",x"0
ce0005",x"0c60000",x"052484b",x"0595959",x"058595b",x"03f3d3d",x"0272325",x"01d1c1c",x"0181c22
",x"1000000",x"1191b24",x"1ff001f",x"1000000",x"1000000",x"0b35566",x"0846e74",x"0ff090f",x"0
ff0713",x"0ff0511",x"0ff030f",x"0ff030f",x"0ff010d",x"0ff010d",x"0fe000c",x"0fe000c",x"0fa000c
",x"0f6000c",x"0ee000b",x"0e2000b",x"0d8000a",x"0d0000a",x"0ce0005",x"0c60000",x"04a3c44",x"
0585356",x"0696f72",x"04a4849",x"02f2d2e",x"01f1d1d",x"01a1d20",x"1000000",x"1000000",x"11
a1c22",x"1000000",x"19a1f37",x"1000000",x"1000000",x"0b98b94",x"0745e6f",x"0f20602",x"0fc0006"
,x"0fa000c",x"0fa000c",x"0f8000c",x"0f6000c",x"0f2000b",x"0f0000b",x"0ea000b",x"0e4000b",x"0
dc000a",x"0d4000a",x"0d0000a",x"0cc0000",x"0bb0501",x"03a353e",x"06d6868",x"09ba1a6",x"061666c
",x"0373437",x"0242221",x"01a1c1e",x"0161928",x"1000000",x"1000000",x"1000000",x"1000000",x"
1000000",x"1000000",x"1ff809f",x"1000000",x"099999c",x"0996473",x"07d6369",x"0ed0d01",x"0
ec1100",x"0ee0000",x"0ea000b",x"0e6000b",x"0e2000b",x"0de000a",x"0d6000a",x"0d00000",x"0ca0e00
",x"0c80600",x"0524047",x"042373b",x"0949594",x"0e6e0de",x"0a9a4a9",x"0474a50",x"0272525",x"01
d1d1e",x"0151924",x"1000000",x"1000000",x"11a1c23",x"1191c22",x"1000000",x"1000000",x"11a1c22"
,x"1203437",x"1000000",x"1000000",x"09cffff",x"0b9adb3",x"09e7480",x"0776b73",x"073434a",x"0
a73319",x"0ce2802",x"0d21a00",x"0c51c01",x"0992517",x"0613c44",x"049414c",x"04e3f41",x"0676465
",x"0a7abaf",x"0dbd9d9",x"0b2adaf",x"05d5a5e",x"01f2022",x"015171c",x"0171b29",x"1000000",x"
1000000",x"11a1c23",x"11a1c23",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"
1000000",x"1f4ffff",x"1000000",x"1000000",x"0b6ded9",x"0bec9cb",x"0b6b8bd",x"0a2a1a7",x"
0868189",x"06b636d",x"05e5861",x"05e5c63",x"0606469",x"06b7074",x"06b7279",x"0677079",x"
0636971",x"052545c",x"03e3e44",x"01b1b1f",x"00f1217",x"121223c",x"1000000",x"1131927",x"11
a1c22",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000"
,x"1000000",x"1ffffff",x"1000000",x"1000000",x"1000000",x"0516870",x"070838a",x"0819299",x"07
```

```
    e8e97",x"074838e",x"0687580",x"0606b76",x"0575f6a",x"0494e58",x"0393b41",x"027282c",x"0121418"
    ,x"000030a",x"1000000",x"1000000",x"1000000",x"1000000",x"11a1c23",x"1000000",x"1000000",x"
    1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"1000000",x"11a1b23",x"
    1000000",x"1000000",x"109070a",x"1000000",x"1000000",x"1000000",x"0000000",x"02c323d",x"
    0313640",x"02e333c",x"02b2f37",x"02a2d35",x"022232b",x"0131418",x"000030e",x"10a1c5f",x"
    1000000",x"1000000",x"1161921",x"11a1c23",x"1000000",x"1000000",x"1000000",x"1000000",x"
    1000000",x"1000000");
51
52
53
54 begin
55
56
57
58
59
60 process (Clk)
61 begin
62 if rising_edge(Clk) then
63 if en = '1' then data <= ROM(TO_INTEGER(addr))(24 downto 0 ); end if;
64 end if;
65 end process;
66
67 end ar;
```

Listing 14: FPGA/scoreController.vhd

```
1 -- This file is part of The Awesome Guitar Game.
2 --
3 --    The Awesome Guitar Game is a free software: you can redistribute it and/or modify
4 --    it under the terms of the GNU General Public License as published by
5 --    the Free Software Foundation, either version 3 of the License, or
6 --    (at your option) any later version.
7 --
8 --    The Awesome Guitar Game is distributed in the hope that it will be useful,
9 --    but WITHOUT ANY WARRANTY; without even the implied warranty of
10 --   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
11 --   GNU General Public License for more details.
12 --
13 --    You should have received a copy of the GNU General Public License
14 --    along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
15 --
16 --
17 -----------------------------------------------------------------------
18 -- The awesome guitar game (A clone of Guitar Hero for FPGA #
19 -----------------------------------------------------------------------
20 -- EMBEDDED SYSTEM PROJECT
21 -- Columbia Unviersity Spring 2012
22 --
23 -- Avijit Singh Wasu -- asw2156@columbia.edu
24 -- Laurent Charignon -- lc2817@columbia.edu
25 --
26 -- Licensed under the GPL license
27 -- Have a look at the license file in the root of
28 -- the project to have more details about the license
29 --
30 -----------------------------------------------------------------------
```

```
31 library ieee;
32 use ieee.std_logic_1164.all;
33 use ieee.numeric_std.all;
34 --This component is in charge of displaying the score
35 --On the 4 rightmost HEX displays
36
37 entity scoreController is
38
39   port (
40     --Avalon
41    clk         : in  std_logic;
42     reset_n     : in  std_logic;
43     read        : in  std_logic;
44     write       : in  std_logic;
45     chipselect  : in  std_logic;
46     address     : in  unsigned(1 downto 0);
47     readdata    : out unsigned(15 downto 0);
48     writedata   : in  unsigned(15 downto 0);
49    --To HEX DISPLAY
50    HEX_0     : out std_logic_vector(6 downto 0);
51    HEX_1 : out std_logic_vector(6 downto 0);
52    HEX_2 : out std_logic_vector(6 downto 0);
53    HEX_3 : out std_logic_vector(6 downto 0)
54    );
55 end scoreController;
56
57 architecture ar of scoreController is
58
59 component hex_decoder is port(
60    di     : in unsigned(3 downto 0);
61    do     : out std_logic_vector( 6 downto 0 )
62 );
63 end component hex_decoder;
64 signal we:std_logic;
65 signal in_val: unsigned (15 downto 0) ;
66 signal d0: unsigned (3 downto 0);
67 signal d1: unsigned (3 downto 0);
68 signal d2: unsigned (3 downto 0);
69 signal d3: unsigned (3 downto 0);
70 begin
71 H0: hex_decoder port map(d0,HEX_0);
72 H1: hex_decoder port map(d1,HEX_1);
73 H2: hex_decoder port map(d2,HEX_2);
74 H3: hex_decoder port map(d3,HEX_3);
75 we <= '1' when chipselect ='1' and write='1' else '0';
76 --in_val <= writedata when we ='1' else in_val;
77 d0 <= in_val(3 downto 0);
78 d1 <= in_val(7 downto 4);
79 d2 <= in_val(11 downto 8);
80 d3 <= in_val(15 downto 12);
81
82
83
84
85 process(clk)
86 begin
87 if(rising_edge(clk)) then
88    in_val <= in_val;
```

```
89    if( reset_n='0' )
90    then
91          in_val <= x"0000";
92    else
93          if (we = '1') then
94             in_val <= writedata ;
95          end if;
96    end if ;
97 end if ;
98 end process ;
99
100 end ar;
```

Listing 15: FPGA/spritecontroller.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity sprite_controller is
6
7    port (
8      VGA_CLK,                              -- Clock
9      VGA_HS,                               -- H_SYNC
10     VGA_VS,                               -- V_SYNC
11     VGA_BLANK,                            -- BLANK
12     VGA_SYNC : out std_logic;             -- SYNC
13     VGA_R,                                -- Red[9:0]
14     VGA_G,                                -- Green[9:0]
15     VGA_B : out unsigned(9 downto 0); -- Blue[9:0]
16      clkin       : in  std_logic;
17      reset_n     : in  std_logic;
18      read        : in  std_logic;
19      write       : in  std_logic;
20      chipselect  : in  std_logic;
21      address     : in  unsigned(8 downto 0);
22      readdata    : out unsigned(15 downto 0);
23      writedata   : in  unsigned(15 downto 0)
24      );
25
26 end sprite_controller;
27
28 architecture rtl of sprite_controller is
29
30
31   constant HTOTAL       : integer := 800;
32   constant HSYNC        : integer := 96;
33   constant HBACK_PORCH  : integer := 48;
34   constant HACTIVE      : integer := 640;
35   constant HFRONT_PORCH : integer := 16;
36
37   constant VTOTAL       : integer := 525;
38   constant VSYNC        : integer := 2;
39   constant VBACK_PORCH  : integer := 33;
40   constant VACTIVE      : integer := 480;
41   constant VFRONT_PORCH : integer := 10;
42
```

```
  signal clk25: std_logic :='0';
  signal clkslow: std_logic :='0';

  signal EndOfLine, EndOfField : std_logic;
  signal data_received : unsigned (11 downto 0);
  signal internal_read : unsigned (11 downto 0);
  signal Hcount : unsigned(9 downto 0);  -- Horizontal position (0-800)
  signal Vcount : unsigned(9 downto 0);  -- Vertical position (0-524)

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

--signal spritno:integer:=0;
--signal x:integer;
--signal yc,xc:integer;
--signal delay:unsigned(31 downto 0);--it has to store values upto 640*480


signal xcoordinate:integer:=0;
signal ycoordinate:integer:=0;
-------------------------------------------------------RAM
    ------------------------------------------------------------------
 type ram_type is array(15 downto 0) of unsigned(14 downto 0); --max of 16 beats can be displayed
     on the screen..X-3 bits,Y-10 bits,correct/wrong-1bit,print or not--Since the X represents the
      sprite no we need to have a value to indicate whether we need to print the sprite or print
     nothin there
 --type Matrix1_type is array(14 downto 0) of ram_type;
 signal RAM : ram_type:=("111111111111111",others=>"000000000000000");
 --type ram_data is array(15 downto 0) of unsigned (14 downto 0);
 signal ram_address:unsigned(3 downto 0);
 type xbutton_type is array(15 downto 0) of integer;
 type ybutton_type is array(15 downto 0) of integer;
 type ritwrong_type is array(15 downto 0) of  std_logic;
 type pixeloutxy_type is array(15 downto 0) of integer;
 type spritno_type is array (15 downto 0) of integer;
type enable_type is array(15 downto 0) of std_logic;
type display_type is array(15 downto 0) of std_logic;

signal xbutton:xbutton_type;
 signal ybutton:ybutton_type;
 signal ritwrong:ritwrong_type ;
 signal enable:enable_type;
signal display:display_type;
signal pixelx:pixeloutxy_type;
signal pixely:pixeloutxy_type;
signal spritno:spritno_type;
signal printspritepixelx:integer:=0;
signal printspritepixely:integer:=0;
signal printsprite:integer:=0;

signal temp0,temp1,temp2,temp3:unsigned (13 downto 0);
signal Xbutton0:integer:=0;
signal Ybutton0:integer:=0;
signal displayoutsum:unsigned(27 downto 0);
signal readfromrom1:unsigned (31 downto 0);
signal readfromrom2:unsigned (31 downto 0);
```

```
signal readfromrom3:unsigned (31 downto 0);
signal readfromrom4:unsigned (31 downto 0);
signal readfromrom5:unsigned (31 downto 0);
signal temp :unsigned(31 downto 0);


signal rom_data:unsigned(31 downto 0);
signal score:integer:=0;
component buttondisplay is
port(Xcoordinate:in integer;
Ycoordinate:in integer;
Xbutton: in integer;
Ybutton: in integer;
display:in std_logic;
ritwrong:in std_logic;
pixeloutx: out integer;--The pixel that need to be displayed
pixelouty: out integer;
spritno:out integer;
enable:out std_logic);

end component;



component rom123 is
port (
Clk : in std_logic;
en : in std_logic; -- Read enable
addrx : in integer;
addry: in integer;
printspritenumber:in integer;
data : out unsigned(31 downto 0)
);
end component;


begin

tryrom1:rom123 port map(Clk =>clkin,en=>'1',addrx=>printspritepixelx,addry=>printspritepixely,
    printspritenumber=>printsprite,data=>readfromrom1 );
tryrom2:rom123 port map(Clk =>clkin,en=>'1',addrx=>printspritepixelx,addry=>printspritepixely,
    printspritenumber=>printsprite,data=>readfromrom2 );
tryrom3:rom123 port map(Clk =>clkin,en=>'1',addrx=>printspritepixelx,addry=>printspritepixely,
    printspritenumber=>printsprite,data=>readfromrom3 );
tryrom4:rom123 port map(Clk =>clkin,en=>'1',addrx=>printspritepixelx,addry=>printspritepixely,
    printspritenumber=>printsprite,data=>readfromrom4 );
tryrom5:rom123 port map(Clk =>clkin,en=>'1',addrx=>printspritepixelx,addry=>printspritepixely,
    printspritenumber=>printsprite,data=>readfromrom5 );


try1:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(0),
    Ybutton=>Ybutton(0),display=>display(0),ritwrong=>ritwrong(0),pixeloutx=>pixelx(0),pixelouty=>
    pixely(0),spritno=>spritno(0),enable=>enable(0));
try2:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(1),
    Ybutton=>Ybutton(1),display=>display(1),ritwrong=>ritwrong(1),pixeloutx=>pixelx(1),pixelouty=>
    pixely(1),spritno=>spritno(1),enable=>enable(1));
try3:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(2),
```

```
Ybutton=>Ybutton(2),display=>display(2),ritwrong=>ritwrong(2),pixeloutx=>pixelx(2),pixelouty=>pixely(2),spritno=>spritno(2),enable=>enable(2));
try4:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(3),Ybutton=>Ybutton(3),display=>display(3),ritwrong=>ritwrong(3),pixeloutx=>pixelx(3),pixelouty=>pixely(3),spritno=>spritno(3),enable=>enable(3));
try5:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(4),Ybutton=>Ybutton(4),display=>display(4),ritwrong=>ritwrong(4),pixeloutx=>pixelx(4),pixelouty=>pixely(4),spritno=>spritno(4),enable=>enable(4));
try6:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(5),Ybutton=>Ybutton(5),display=>display(5),ritwrong=>ritwrong(5),pixeloutx=>pixelx(5),pixelouty=>pixely(5),spritno=>spritno(5),enable=>enable(5));
try7:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(6),Ybutton=>Ybutton(6),display=>display(6),ritwrong=>ritwrong(6),pixeloutx=>pixelx(6),pixelouty=>pixely(6),spritno=>spritno(6),enable=>enable(6));
try8:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(7),Ybutton=>Ybutton(7),display=>display(7),ritwrong=>ritwrong(7),pixeloutx=>pixelx(7),pixelouty=>pixely(7),spritno=>spritno(7),enable=>enable(7));
try9:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(8),Ybutton=>Ybutton(8),display=>display(8),ritwrong=>ritwrong(8),pixeloutx=>pixelx(8),pixelouty=>pixely(8),spritno=>spritno(8),enable=>enable(8));
try10:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(9),Ybutton=>Ybutton(9),display=>display(9),ritwrong=>ritwrong(9),pixeloutx=>pixelx(9),pixelouty=>pixely(9),spritno=>spritno(9),enable=>enable(9));
try11:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(10),Ybutton=>Ybutton(10),display=>display(10),ritwrong=>ritwrong(10),pixeloutx=>pixelx(10),pixelouty=>pixely(10),spritno=>spritno(10),enable=>enable(10));
try12:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(11),Ybutton=>Ybutton(11),display=>display(11),ritwrong=>ritwrong(11),pixeloutx=>pixelx(11),pixelouty=>pixely(11),spritno=>spritno(11),enable=>enable(11));
try13:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(12),Ybutton=>Ybutton(12),display=>display(12),ritwrong=>ritwrong(12),pixeloutx=>pixelx(12),pixelouty=>pixely(12),spritno=>spritno(12),enable=>enable(12));
try14:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(13),Ybutton=>Ybutton(13),display=>display(13),ritwrong=>ritwrong(13),pixeloutx=>pixelx(13),pixelouty=>pixely(13),spritno=>spritno(13),enable=>enable(13));
try15:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(14),Ybutton=>Ybutton(14),display=>display(14),ritwrong=>ritwrong(14),pixeloutx=>pixelx(14),pixelouty=>pixely(14),spritno=>spritno(14),enable=>enable(14));
try16:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton(15),Ybutton=>Ybutton(15),display=>display(15),ritwrong=>ritwrong(15),pixeloutx=>pixelx(15),pixelouty=>pixely(15),spritno=>spritno(15),enable=>enable(15));

--try4:buttondisplay port map(Xcoordinate=>Xcoordinate,Ycoordinate=>Ycoordinate,Xbutton=>Xbutton3,Ybutton=>Ybutton3,pixelout=>displayout3);



--displayoutsum<=displayout(0)+displayout(1)+displayout(2)+displayout(3)+displayout(4)+displayout(5)+displayout(6)+displayout(7)+displayout(8)+displayout(9)+displayout(10)+displayout(11)+displayout(12)+displayout(13)+displayout(14)+displayout(15);


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

printsprite <= spritno(0) when enable(0) = '1'else
spritno(1) when enable(1) = '1' else
spritno(2) when enable(2) = '1' else
```

```
172 spritno(3) when enable(3) = '1' else
173 spritno(4) when enable(4) = '1' else
174 spritno(5) when enable(5) = '1' else
175 spritno(6) when enable(6) = '1' else
176 spritno(7) when enable(7) = '1' else
177 spritno(8) when enable(8) = '1' else
178 spritno(9) when enable(9) = '1' else
179 spritno(10) when enable(10) = '1' else
180 spritno(11) when enable(11) = '1' else
181 spritno(12) when enable(12) = '1' else
182 spritno(13) when enable(13) = '1' else
183 spritno(14) when enable(14) ='1' else
184 spritno(15) when enable(15) = '1' else
185 0;
186
187 printspritepixelx <= pixelx(0) when enable(0) = '1'else
188 pixelx(1) when enable(1) = '1' else
189 pixelx(2) when enable(2) = '1' else
190 pixelx(3) when enable(3) = '1' else
191 pixelx(4) when enable(4) = '1' else
192 pixelx(5) when enable(5) = '1' else
193 pixelx(6) when enable(6) = '1' else
194 pixelx(7) when enable(7) = '1' else
195 pixelx(8) when enable(8) = '1' else
196 pixelx(9) when enable(9) = '1' else
197 pixelx(10) when enable(10) = '1' else
198 pixelx(11) when enable(11) = '1' else
199 pixelx(12) when enable(12) = '1' else
200 pixelx(13) when enable(13) = '1' else
201 pixelx(14) when enable(14) ='1' else
202 pixelx(15) when enable(15) = '1' else
203 0;
204
205 printspritepixely <= pixely(0) when enable(0) = '1'else
206 pixely(1) when enable(1) = '1' else
207 pixely(2) when enable(2) = '1' else
208 pixely(3) when enable(3) = '1' else
209 pixely(4) when enable(4) = '1' else
210 pixely(5) when enable(5) = '1' else
211 pixely(6) when enable(6) = '1' else
212 pixely(7) when enable(7) = '1' else
213 pixely(8) when enable(8) = '1' else
214 pixely(9) when enable(9) = '1' else
215 pixely(10) when enable(10) = '1' else
216 pixely(11) when enable(11) = '1' else
217 pixely(12) when enable(12) = '1' else
218 pixely(13) when enable(13) = '1' else
219 pixely(14) when enable(14) ='1' else
220 pixely(15) when enable(15) = '1' else
221 0;
222
223
224
225
226
227
228
229 -- Horizontal and vertical counters
```

```
230 HCounter : process (clk25)
231  begin
232    if rising_edge(clk25) then
233      if reset = '1' then
234        Hcount <= (others => '0');
235      elsif EndOfLine = '1' then
236        Hcount <= (others => '0');
237      else
238        Hcount <= Hcount + 1;
239      end if;
240    end if;
241
242  end process HCounter;
243
244  EndOfLine <= '1' when Hcount = HTOTAL - 1 else '0';
245
246  VCounter: process (clk25)
247  begin
248    if rising_edge(clk25) then
249      if reset = '1' then
250        Vcount <= (others => '0');
251      elsif EndOfLine = '1' then
252        if EndOfField = '1' then
253          Vcount <= (others => '0');
254        else
255          Vcount <= Vcount + 1;
256        end if;
257      end if;
258    end if;
259  end process VCounter;
260
261  EndOfField <= '1' when Vcount = VTOTAL - 1 else '0';
262
263
264
265 ---------------------------------- Registered video signals going to the video DAC
       ----------------------------------------
266
267
268
269  ---------------------------------- State machines to generate HSYNC, VSYNC, HBLANK, and VBLANK
       ----------------------------------
270
271  HSyncGen : process (clk25)
272  begin
273    if rising_edge(clk25) then
274      if reset = '1' or EndOfLine = '1' then
275        vga_hsync <= '1';
276      elsif Hcount = HSYNC - 1 then
277        vga_hsync <= '0';
278      end if;
279    end if;
280  end process HSyncGen;
281
282  HBlankGen : process (clk25)
283  begin
284    if rising_edge(clk25) then
285      if reset = '1' then
```

```
286         vga_hblank <= '1';
287       elsif Hcount = HSYNC + HBACK_PORCH then
288         vga_hblank <= '0';
289       elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE then
290         vga_hblank <= '1';
291       end if;
292     end if;
293   end process HBlankGen;
294
295   VSyncGen : process (clk25)
296   begin
297     if rising_edge(clk25) then
298       if reset = '1' then
299         vga_vsync <= '1';
300       elsif EndOfLine ='1' then
301         if EndOfField = '1' then
302           vga_vsync <= '1';
303         elsif Vcount = VSYNC - 1 then
304           vga_vsync <= '0';
305         end if;
306       end if;
307     end if;
308   end process VSyncGen;
309
310   VBlankGen : process (clk25)
311   begin
312     if rising_edge(clk25) then
313       if reset = '1' then
314         vga_vblank <= '1';
315       elsif EndOfLine = '1' then
316         if Vcount = VSYNC + VBACK_PORCH - 1 then
317           vga_vblank <= '0';
318         elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
319           vga_vblank <= '1';
320         end if;
321       end if;
322     end if;
323   end process VBlankGen;
324
325
326 --
327
328 ------------------------------------------------------------------------------------------------------------------------

329  Xcoordinate<=(to_integer(Hcount)-HSYNC - HBACK_PORCH);
330  Ycoordinate<=(to_integer(Vcount)-VSYNC - VBACK_PORCH);
331
332
333 ------------------------------EXCTRACTING DATA FROM RAM
      ------------------------------------------------------
334   reset <= not reset_n;
335    ram_address<= address(3 downto 0) ;--ram_adress is 6 bits
336
337
338   process (clkin)       ------Process for reading/writing data from the RAM based on chipselect and
        reset_n
339   begin
340     if rising_edge(clkin) then
```

```
341       if reset_n = '0' then
342         readdata <= (others => '0');
343       else
344         if chipselect = '1' then
345             if read = '1' then
346               --readdata <= "00" & RAM(to_integer(ram_address));
347                --readdata <= "0001011111111111" ;
348 readdata <= "0" & RAM(to_integer(ram_address));
349 --readdata <=rom_data(15 downto 0);
350 temp<=X"0ff3fc00";
351 --readdata <= "000000000000000" & display(6);
352 --readdata <= "00000000" & (ram_data(7 downto 0));
353                 elsif write = '1' then
354                 RAM(to_integer(ram_address)) <= writedata(15 downto 0);
355 
356 
357                 end if;
358             else
359 
360 
361 end if;
362 
363 
364  end if;
365  end if;
366  end process;
367 
368 
369 --Xbutton(0)<=to_integer(RAM(0)(2 downto 0));
370 --Ybutton(0)<=to_integer(RAM(0)(12 downto 3));
371 --display(0)<=(RAM(0)(13));
372 --ritwrong(0)<=(RAM(0)(14));
373 
374 Xbutton(1)<=to_integer(RAM(1)(2 downto 0));
375 Ybutton(1)<=to_integer(RAM(1)(12 downto 3));
376 display(1)<=(RAM(1)(13));
377 ritwrong(1)<=(RAM(1)(14));
378 
379 Xbutton(2)<=to_integer(RAM(2)(2 downto 0));
380 Ybutton(2)<=to_integer(RAM(2)(12 downto 3));
381 display(2)<=(RAM(2)(13));
382 ritwrong(2)<=(RAM(2)(14));
383 
384 Xbutton(3)<=to_integer(RAM(3)(2 downto 0));
385 Ybutton(3)<=to_integer(RAM(3)(12 downto 3));
386 display(3)<=(RAM(3)(13));
387 ritwrong(3)<=(RAM(3)(14));
388 
389 Xbutton(4)<=to_integer(RAM(4)(2 downto 0));
390 Ybutton(4)<=to_integer(RAM(4)(12 downto 3));
391 display(4)<=(RAM(4)(13));
392 ritwrong(4)<=(RAM(4)(14));
393 
394 
395 Xbutton(5)<=to_integer(RAM(5)(2 downto 0));
396 Ybutton(5)<=to_integer(RAM(5)(12 downto 3));
397 display(5)<=(RAM(5)(13));
398 ritwrong(5)<=(RAM(5)(14));
```

```
399
400
401 Xbutton(6)<=to_integer(RAM(6)(2 downto 0));
402 Ybutton(6)<=to_integer(RAM(6)(12 downto 3));
403 display(6)<=(RAM(6)(13));
404 ritwrong(6)<=(RAM(6)(14));
405
406
407 Xbutton(7)<=to_integer(RAM(7)(2 downto 0));
408 Ybutton(7)<=to_integer(RAM(7)(12 downto 3));
409 display(7)<=(RAM(7)(13));
410 ritwrong(7)<=(RAM(7)(14));
411
412
413 Xbutton(8)<=to_integer(RAM(8)(2 downto 0));
414 Ybutton(8)<=to_integer(RAM(8)(12 downto 3));
415 display(8)<=(RAM(8)(13));
416 ritwrong(8)<=(RAM(8)(14));
417
418 Xbutton(9)<=to_integer(RAM(9)(2 downto 0));
419 Ybutton(9)<=to_integer(RAM(9)(12 downto 3));
420 display(9)<=(RAM(9)(13));
421 ritwrong(9)<=(RAM(9)(14));
422
423 Xbutton(10)<=to_integer(RAM(10)(2 downto 0));
424 Ybutton(10)<=to_integer(RAM(10)(12 downto 3));
425 display(10)<=(RAM(10)(13));
426 ritwrong(10)<=(RAM(10)(14));
427
428 Xbutton(11)<=to_integer(RAM(11)(2 downto 0));
429 Ybutton(11)<=to_integer(RAM(11)(12 downto 3));
430 display(11)<=(RAM(11)(13));
431 ritwrong(11)<=(RAM(11)(14));
432
433 Xbutton(12)<=to_integer(RAM(12)(2 downto 0));
434 Ybutton(12)<=to_integer(RAM(12)(12 downto 3));
435 display(12)<=(RAM(12)(13));
436 ritwrong(12)<=(RAM(12)(14));
437
438 Xbutton(13)<=to_integer(RAM(13)(2 downto 0));
439 Ybutton(13)<=to_integer(RAM(13)(12 downto 3));
440 display(13)<=(RAM(13)(13));
441 ritwrong(13)<=(RAM(13)(14));
442
443 Xbutton(14)<=to_integer(RAM(14)(2 downto 0));
444 Ybutton(14)<=to_integer(RAM(14)(12 downto 3));
445 display(14)<=(RAM(14)(13));
446 ritwrong(14)<=(RAM(14)(14));
447
448 Xbutton(15)<=to_integer(RAM(15)(2 downto 0));
449 Ybutton(15)<=to_integer(RAM(15)(12 downto 3));
450 display(15)<=(RAM(15)(13));
451 ritwrong(15)<=(RAM(15)(14));
452
453 score<=to_integer(RAM(0));
454
455
456
```

```
457 ------------------------------------------------------------generating clock for the VGA
        ------------------------------------------
458  process (clkin)
459   begin
460     if rising_edge(clkin) then
461       clk25 <= not clk25;
462     end if;
463  end process;
464 ------------------------------------------------ASSIGNING THE RAM DATA TO THE VARIABLES
        ------------------------------------------
465
466
467 ------------------------------------------------dealing with sprite offset
        ---------------------------------------
468
469
470
471
472
473
474
475 process (clk25)      ------Process for reading data from the ROM based on reset_n
        ??????????????????????????????
476  begin
477
478 if rising_edge(clk25) then
479 if reset_n = '0' then
480 rom_data<=X"00000000";
481
482 else if (printsprite=0 and Xcoordinate>140 and Xcoordinate<370 and Ycoordinate>380 and Ycoordinate
        <400) then
483 rom_data<=X"0ff3fc00";
484
485 else if (printsprite=0 and Xcoordinate=156)or (printsprite=0 and Xcoordinate=156+50) or (
        printsprite=0 and Xcoordinate=156+100) or (printsprite=0 and Xcoordinate=156+150) or (
        printsprite=0 and Xcoordinate=156+200) then --displaying the white lines
486 rom_data<=X"ffffffff";
487
488 else if (printsprite=0 and (Xcoordinate-475)*(Xcoordinate-475)+(Ycoordinate-75)*(Ycoordinate-75)
        <625 and score>0        ) then
489 rom_data<=X"f1f1f1f1";
490 else if (printsprite=0 and (Xcoordinate-525)*(Xcoordinate-525)+(Ycoordinate-80)*(Ycoordinate-80)
        <400 and score>1) then
491 rom_data<=X"f1f1f1f1";
492 else if (printsprite=0 and (Xcoordinate-565)*(Xcoordinate-565)+(Ycoordinate-85)*(Ycoordinate-85)
        <225 and score>2) then
493 rom_data<=X"f1f1f1f1";
494 else if (printsprite=0 and (Xcoordinate-595)*(Xcoordinate-595)+(Ycoordinate-90)*(Ycoordinate-90)
        <100 and score>3) then
495 rom_data<=X"f1f1f1f1";
496 else if (printsprite=0 and (Xcoordinate-615)*(Xcoordinate-615)+(Ycoordinate-95)*(Ycoordinate-95)
        <25 and score>4) then
497 rom_data<=X"f1f1f1f1";
498
499 else if (printsprite=0 and Xcoordinate>140 and Xcoordinate<370 and Ycoordinate>380 and Ycoordinate
        <400) then
500 rom_data<=X"0ff3fc00";
501
```

```
502 else if (printsprite=0 and Xcoordinate=156)or (printsprite=0 and Xcoordinate=156+50) or (
        printsprite=0 and Xcoordinate=156+100) or (printsprite=0 and Xcoordinate=156+150) or (
        printsprite=0 and Xcoordinate=156+200) then --displaying the white lines
503 rom_data<=X"ffffffff";
504
505 else if (printsprite=0) then
506 rom_data<=X"00000000";
507
508 else  if (printsprite=1) then
509 rom_data <= readfromrom1;
510
511 else if (printsprite=2) then
512 --rom_data <= Ma2(printspritepixelx)(printspritepixely);
513 rom_data <= readfromrom1;
514
515 else if (printsprite=3) then
516 --rom_data <= Ma2(printspritepixelx)(printspritepixely);
517 rom_data <= readfromrom1;
518
519 else if (printsprite=4) then
520 --rom_data <= Ma1(printspritepixelx)(printspritepixely);
521 rom_data <= readfromrom2;
522
523 else if (printsprite=5) then
524 --rom_data <= Ma1(printspritepixelx)(printspritepixely);
525 rom_data <= readfromrom3;
526
527 else if (printsprite=6) then
528 --rom_data <= Ma1(printspritepixelx)(printspritepixely);
529 rom_data <= readfromrom4;
530 else if (printsprite=7) then
531 --rom_data <= Ma1(printspritepixelx)(printspritepixely);
532 rom_data <= readfromrom5;
533
534 else rom_data<=X"00000000";
535
536 end if;
537 end if;
538 end if;
539 end if;
540 end if;
541 end if;
542 end if;
543 end if;
544 end if;
545 end if;
546 end if;
547 end if;
548 end if;
549 end if;
550 end if;
551 end if;
552 end if;
553 end if;
554 end if;
555 end process;
556 --
557 ----readdata <= x"0" & internal_read (11 downto 0);
```

```
----data_received <= writedata (11 downto 0);



VideoOut: process (clk25, reset)
begin
if reset = '1' then

VGA_R <= "0000000000" ;
VGA_G <= "0000000000"  ;--& rom_data(17 downto 9);
VGA_B <= "0000000000";-- & rom_data(26 downto 18);--"1111111111";



else if clk25'event and clk25 = '1' then
   if vga_hblank='0' and vga_vblank='0' then
   if(printsprite>0 and Xcoordinate>140 and Xcoordinate<370 and Ycoordinate>380 and Ycoordinate
       <400 and rom_data(29 downto 0)=0)then
        VGA_G <=temp  (19 downto 10);
   VGA_R <= temp(9 downto 0);
   VGA_B <=  temp(29 downto 20);--"1111111111";
    else
   VGA_G <=  rom_data(19 downto 10);
   VGA_R <= rom_data(9 downto 0);
   VGA_B <=  rom_data(29 downto 20);--"1111111111";
end if;


else
     VGA_R <= "0000000000";
     VGA_G <= "0000000000";
    VGA_B <= "0000000000";
   end if;
   end if;
end if;
 end process VideoOut;

VGA_CLK <= clk25;
VGA_HS <= not vga_hsync;
VGA_VS <= not vga_vsync;
VGA_SYNC <= '0';
VGA_BLANK <= not (vga_hsync or vga_vsync);

end rtl;
--
```

Listing 16: FPGA/timer.vhd

```
--Legal Notice: (C)2007 Altera Corporation. All rights reserved.  Your
--use of Altera Corporation's design tools, logic functions and other
--software and tools, and its AMPP partner logic functions, and any
--output files any of the foregoing (including device programming or
--simulation files), and any associated documentation or information are
--expressly subject to the terms and conditions of the Altera Program
--License Subscription Agreement or other applicable license agreement,
--including, without limitation, that your use is for the sole purpose
```

```
--of programming logic devices manufactured by Altera and sold by Altera
--or its authorized distributors.  Please refer to the applicable
--agreement for further details.


-- turn off superfluous VHDL processor warnings
-- altera message_level Level1
-- altera message_off 10034 10035 10036 10037 10230 10240 10030

library altera;
use altera.altera_europa_support_lib.all;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity timer is
        port (
              -- inputs:
                 signal address : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
                 signal chipselect : IN STD_LOGIC;
                 signal clk : IN STD_LOGIC;
                 signal reset_n : IN STD_LOGIC;
                 signal write_n : IN STD_LOGIC;
                 signal writedata : IN STD_LOGIC_VECTOR (15 DOWNTO 0);

              -- outputs:
                 signal irq : OUT STD_LOGIC;
                 signal readdata : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
              );
end entity timer;


architecture europa of timer is
                signal clk_en :  STD_LOGIC;
                signal control_continuous :  STD_LOGIC;
                signal control_interrupt_enable :  STD_LOGIC;
                signal control_register :  STD_LOGIC_VECTOR (3 DOWNTO 0);
                signal control_wr_strobe :  STD_LOGIC;
                signal counter_is_running :  STD_LOGIC;
                signal counter_is_zero :  STD_LOGIC;
                signal counter_load_value :  STD_LOGIC_VECTOR (18 DOWNTO 0);
                signal delayed_unxcounter_is_zeroxx0 :  STD_LOGIC;
                signal do_start_counter :  STD_LOGIC;
                signal do_stop_counter :  STD_LOGIC;
                signal force_reload :  STD_LOGIC;
                signal internal_counter :  STD_LOGIC_VECTOR (18 DOWNTO 0);
                signal period_h_wr_strobe :  STD_LOGIC;
                signal period_l_wr_strobe :  STD_LOGIC;
                signal read_mux_out :  STD_LOGIC_VECTOR (15 DOWNTO 0);
                signal start_strobe :  STD_LOGIC;
                signal status_wr_strobe :  STD_LOGIC;
                signal stop_strobe :  STD_LOGIC;
                signal timeout_event :  STD_LOGIC;
                signal timeout_occurred :  STD_LOGIC;

begin
```

```
67
68  clk_en <= std_logic'('1');
69  process (clk, reset_n)
70  begin
71    if reset_n = '0' then
72      internal_counter <= std_logic_vector'("11110100001000111111");
73    elsif clk'event and clk = '1' then
74      if std_logic'((counter_is_running OR force_reload)) = '1' then
75        if std_logic'((counter_is_zero OR force_reload)) = '1' then
76          internal_counter <= counter_load_value;
77        else
78          internal_counter <= A_EXT (((std_logic_vector'("00000000000000") & (internal_counter)) -
                std_logic_vector'("00000000000000000000000000000001")), 19);
79        end if;
80      end if;
81    end if;
82
83  end process;
84
85  counter_is_zero <= to_std_logic(((std_logic_vector'("0000000000000") & (internal_counter)) =
      std_logic_vector'("00000000000000000000000000000000")));
86  counter_load_value <= std_logic_vector'("11110100001000111111");
87  process (clk, reset_n)
88  begin
89    if reset_n = '0' then
90      force_reload <= std_logic'('0');
91    elsif clk'event and clk = '1' then
92      if std_logic'(clk_en) = '1' then
93        force_reload <= period_h_wr_strobe OR period_l_wr_strobe;
94      end if;
95    end if;
96
97  end process;
98
99  do_start_counter <= start_strobe;
100 do_stop_counter <= ((stop_strobe) OR (force_reload)) OR ((counter_is_zero AND NOT
      control_continuous));
101 process (clk, reset_n)
102 begin
103   if reset_n = '0' then
104     counter_is_running <= std_logic'('0');
105   elsif clk'event and clk = '1' then
106     if std_logic'(clk_en) = '1' then
107       if std_logic'(do_start_counter) = '1' then
108         counter_is_running <= Vector_To_Std_Logic(-SIGNED(std_logic_vector'
              ("00000000000000000000000000000001")));
109       elsif std_logic'(do_stop_counter) = '1' then
110         counter_is_running <= std_logic'('0');
111       end if;
112     end if;
113   end if;
114
115 end process;
116
117 --delayed_unxcounter_is_zeroxx0, which is an e_register
118 process (clk, reset_n)
119 begin
120   if reset_n = '0' then
```

```
121       delayed_unxcounter_is_zeroxx0 <= std_logic'('0');
122     elsif clk'event and clk = '1' then
123       if std_logic'(clk_en) = '1' then
124         delayed_unxcounter_is_zeroxx0 <= counter_is_zero;
125       end if;
126     end if;
127
128   end process;
129
130   timeout_event <= (counter_is_zero) AND NOT (delayed_unxcounter_is_zeroxx0);
131   process (clk, reset_n)
132   begin
133     if reset_n = '0' then
134       timeout_occurred <= std_logic'('0');
135     elsif clk'event and clk = '1' then
136       if std_logic'(clk_en) = '1' then
137         if std_logic'(status_wr_strobe) = '1' then
138           timeout_occurred <= std_logic'('0');
139         elsif std_logic'(timeout_event) = '1' then
140           timeout_occurred <= Vector_To_Std_Logic(-SIGNED(std_logic_vector'
                ("00000000000000000000000000000001")));
141         end if;
142       end if;
143     end if;
144
145   end process;
146
147   irq <= timeout_occurred AND control_interrupt_enable;
148   --s1, which is an e_avalon_slave
149   read_mux_out <= ((A_REP(to_std_logic((((std_logic_vector'("00000000000000000000000000000000") & (
        address)) = std_logic_vector'("00000000000000000000000000000001")))), 16) AND (
        std_logic_vector'("0000000000000") & (control_register)))) OR ((A_REP(to_std_logic((((
        std_logic_vector'("00000000000000000000000000000000") & (address)) = std_logic_vector'("
        00000000000000000000000000000000")))), 16) AND (std_logic_vector'("00000000000000") & (
        Std_Logic_Vector'(A_ToStdLogicVector(counter_is_running) & A_ToStdLogicVector(
        timeout_occurred))))));
150   process (clk, reset_n)
151   begin
152     if reset_n = '0' then
153       readdata <= std_logic_vector'("0000000000000000");
154     elsif clk'event and clk = '1' then
155       if std_logic'(clk_en) = '1' then
156         readdata <= read_mux_out;
157       end if;
158     end if;
159
160   end process;
161
162   period_l_wr_strobe <= (chipselect AND NOT write_n) AND to_std_logic((((std_logic_vector'("
        00000000000000000000000000000000") & (address)) = std_logic_vector'
        ("00000000000000000000000000000010"))));
163   period_h_wr_strobe <= (chipselect AND NOT write_n) AND to_std_logic((((std_logic_vector'("
        00000000000000000000000000000000") & (address)) = std_logic_vector'
        ("00000000000000000000000000000011"))));
164   control_wr_strobe <= (chipselect AND NOT write_n) AND to_std_logic((((std_logic_vector'("
        00000000000000000000000000000000") & (address)) = std_logic_vector'
        ("00000000000000000000000000000001"))));
165   process (clk, reset_n)
```

```
166   begin
167     if reset_n = '0' then
168       control_register <= std_logic_vector'("0000");
169     elsif clk'event and clk = '1' then
170       if std_logic'(control_wr_strobe) = '1' then
171         control_register <= writedata(3 DOWNTO 0);
172       end if;
173     end if;
174
175   end process;
176
177   stop_strobe <= writedata(3) AND control_wr_strobe;
178   start_strobe <= writedata(2) AND control_wr_strobe;
179   control_continuous <= control_register(1);
180   control_interrupt_enable <= control_register(0);
181   status_wr_strobe <= (chipselect AND NOT write_n) AND to_std_logic((((std_logic_vector'
        ("00000000000000000000000000000000") & (address)) = std_logic_vector'("
        00000000000000000000000000000000"))));
182
183 end europa;
```

Listing 17: FPGA/software/MusicPlays/helloworld.c

```
1  // This file is part of The Awesome Guitar Game.
2  //
3  //     The Awesome Guitar Game is a free software: you can redistribute it and/or modify
4  //     it under the terms of the GNU General Public License as published by
5  //     the Free Software Foundation, either version 3 of the License, or
6  //     (at your option) any later version.
7  //
8  //     The Awesome Guitar Game is distributed in the hope that it will be useful,
9  //     but WITHOUT ANY WARRANTY; without even the implied warranty of
10 //     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
11 //     GNU General Public License for more details.
12 //
13 //     You should have received a copy of the GNU General Public License
14 //     along with The Awesome Guitar Game.  If not, see <http://www.gnu.org/licenses/>.
15 //
16 //
17 //#########################################################################
18 // The awesome guitar game (A clone of Guitar Hero for FPGA #
19 //#########################################################################
20 // EMBEDDED SYSTEM PROJECT
21 // Columbia Unviersity Spring 2012
22 //
23 // Avijit Singh Wasu -- asw2156@columbia.edu
24 // Laurent Charignon -- lc2817@columbia.edu
25 //
26 // Licensed under the GPL license
27 // Have a look at the license file in the root of
28 // the project to have more details about the license
29 //
30 //#########################################################################
31
32 #include <stdio.h>
33 #include <system.h>
34 #include <math.h>
```

```
35 #include<io.h>
36 #include <sys/alt_irq.h>
37
38
39 /*******************
40 ** DEFINES
41 ********************/
42 #define REPRBREAT(beat) ((beat.color)+(beat.Y) << 3);
43 #define READNEWNOTE(address)  IORD_8DIRECT(CFI_FLASH_BASE, address);
44 #define READNEWBEAT(current_note)  IORD_16DIRECT(BEATCONTROLLER_INST_BASE, current_note*2)
45 #define WRITENOTE(note)  IOWR_8DIRECT( MUSICCONTROLLER_INST_BASE, 0, (note-127)&0xFF);
46 #define WRITEBEAT(v) IOWR_16DIRECT(VGA_BASE,v*2,(beats[v]));
47 #define NUMCELL 16
48 #define PACE 2
49 #define MAX_Y 480
50
51 /*****************
52  * STRUCTS
53  *****************/
54 //A beat
55 typedef struct beat {
56   unsigned char color;     // Color between 2 and 6 for notes
57   int Y;                   // Y value of the note
58 } beat ;
59
60 /*****************
61  ** ISR CALLBACKS
62  *****************/
63 // We have tree types of callbacks
64 static void  note_isr ( void* context, alt_u32 id);
65 static void  input_isr ( void* context, alt_u32 id);
66 static void  timer_isr ( void* context, alt_u32 id);
67
68 /***********************
69  **GENERAL PROTOTYPES
70  ***********************/
71 void init_sys();
72 void vga_reset();                                    //TODO determine if it is needed
73 inline int get_rand_1_6();
74
75 /************************
76 ** GLOBAL VARIABLES
77 ************************/
78 volatile int current_beat_number  = 0;               //Current note: 40, TODO: compute it instead
79 volatile int next_beat            = 0;
80 volatile long time                = 0;              //Start time for the beats, it has to be the
81 volatile int UP                   = 0;                //time it takes to cross the screen
82 volatile int NEWBEAT              = 0 ;
83 volatile long notes_address       = 0;
84 volatile int next_sample          = 0;
85 volatile int numballs             = 5;
86 volatile int numpressed           = 2 ;
87 volatile int totalballs           = 0 ;
88 volatile int scoretrigger         = 0;
89
90
91  #define threshold  100
92 //TODO CONSTANT
```

```
93
94  int rbase               = 480;//TOBEDEFINED
95  int hbase               = 380-threshold;
96 volatile int score                 = 0;
97 volatile int string = 0;
98 volatile int should_fetch_new_sample = 1;
99
100 /**********************
101  ** IMPLEMENTATION
102  **********************/
103 // Sends the new notes to the music controller
104 // @trigger: 16 000 times per second
105 static void  note_isr ( void* context, alt_u32 id){
106   static int count =0  ;
107   // This whole thing plays one note every two interrupts
108   count ++;
109   if (count == 1){
110     notes_address +=2;count = 0;
111
112   }
113   // Write the note thus cleaning the exception
114   should_fetch_new_sample = 1;
115   WRITENOTE(next_sample);
116
117   return;
118 }
119
120 // Keeps track of user's input
121 // @trigger: when a key is pressed
122 static void  input_isr ( void* context, alt_u32 id){
123   int col =   IORD_16DIRECT( INPUTCONTROLLER_INST_BASE, 0);
124   if (numpressed > 0)
125     numpressed --;
126   switch(col){
127   case 1:
128   string = 2;
129   break;
130   case 2:
131   string = 3;
132   break;
133   case 4:
134   string = 4;
135   break;
136   case 8:
137   string = 5;
138   break;
139   case 16:
140   string = 6;
141   break;
142   default:
143   break;
144   }
145
146   IOWR_8DIRECT( INPUTCONTROLLER_INST_BASE, 0, 0);
147
148
149
150   //TODO implement the score computation
```

```
151   /* HOW TO USE THE HEX DISPLAY
152       if ((current_note&0x2) == 2)
153       IOWR_16DIRECT(SCORECONTROLLER_INST_BASE,0,0xdead);
154       else
155       IOWR_16DIRECT(SCORECONTROLLER_INST_BASE,0,0xbeef);
156
157 */
158   return;
159 }
160
161
162 // Synchronize the timing of the whole system, keeps track of the ellapsed time
163 // @trigger: 100 times per second
164 static void  timer_isr ( void* context, alt_u32 id){
165   time ++;
166   //printf("%d NEXT %d beat_number %d\n",time,next_beat,current_beat_number);
167
168   //UP = 1 means enable a function in the main loop to move
169   //the beats down one step
170   UP = 1;
171   if(scoretrigger == 0 && time > 1500)
172     scoretrigger =1;
173
174   //If a new beat shows up
175   if (next_beat == time){
176     current_beat_number++;
177     NEWBEAT = 1;
178   }
179
180   //Clear the exception
181   IOWR_16DIRECT( TIMER_BASE, 0, 0);
182   return;
183 }
184
185
186
187 inline int get_rand_1_6(){
188         int shifter=0 ;
189         do{
190           shifter = (rand()&0x7);
191         } while (shifter == 7 || (shifter < 1)  );
192
193         return shifter;
194 }
195
196 int main()
197 {
198 if(380+threshold > 480)
199  rbase              = 480;
200 else
201  rbase              = 380+threshold;
202
203
204
205   int i = 0;
206   int beats[NUMCELL];
207   for ( i = 0 ; i<NUMCELL ; i ++){
208   IOWR_16DIRECT(VGA_BASE,i*2,0);
```

```
209   beats[i] = 0;
210 }
211
212
213
214
215 alt_irq_register(INPUTCONTROLLER_INST_IRQ, (void*)0, input_isr);
216 while(numpressed > 0);
217
218
219   //TODO MOVE
220   IOWR_16DIRECT(TIMER_BASE, 2, 0x7);
221   alt_irq_register(TIMER_IRQ, (void*)0, timer_isr);
222   alt_irq_register(MUSICCONTROLLER_INST_IRQ , (void*)0, note_isr);
223
224   //printf("%d %d %d\n",READNEWBEAT(0),READNEWBEAT(1),READNEWBEAT(2));
225   //the cursors of the ARRAY
226   int low = 1;
227   int high = 1;
228   int v = 0;
229   int should_fetch_new_beat = 1;
230   int u ;
231   int pos = 0 ;
232   //the array that stores the notes
233
234   for(;;){
235       IOWR_16DIRECT(SCORECONTROLLER_INST_BASE,0,score);
236 IOWR_16DIRECT(VGA_BASE,0,numballs);
237     if(string > 0){
238
239             u = low;
240             printf("TREATING %d %d\n",low,high);
241             //printf("DEALING with strings %d \n",string);
242             while (u !=high){
243                 pos = (beats[u]>>3) ;
244                 printf("color: %d %d %d %d\n",(beats[u]&0x7), (beats[u]>>3),hbase,rbase);
245                 if( ((beats[u]&0x7) == string) && (hbase<pos) && (rbase>pos) ){
246                     beats[u]= 0 ;
247                     if (u == low){
248                         low ++;
249                         if ( low == NUMCELL)
250                             low = 1;
251                     }
252                     printf("gotcha !\n");
253                     score ++;
254                     if(scoretrigger ==1)
255                         numballs = ((5*score)/totalballs+1);
256
257                     break;
258             }
259       //Optimization ...
260               /* if((beats[u]>>3)<hbase){
261                  printf("too late !\n");
262                     break;
263                 }*/
264
265              u++;
266              if (u == NUMCELL)
```

```
267                u = 1;
268               }
269                     string = 0;
270          }
271
272
273
274
275      if(should_fetch_new_beat == 1){
276
277      next_beat         = READNEWBEAT(current_beat_number);
278      should_fetch_new_beat =0 ;
279      }
280
281      if(should_fetch_new_sample == 1){
282      next_sample = READNEWNOTE(notes_address);
283      should_fetch_new_sample = 0;
284      }
285
286
287      if(UP){
288        // 1) Process new beats
289        if(NEWBEAT){
290       //   printf("at %d = %d \n",v,beats[v]);
291          beats[high] = get_rand_1_6();
292          high++;
293          should_fetch_new_beat =1 ;
294
295          if (high == NUMCELL)     high= 1 ;
296        }
297          v            = low;
298        //2) Move the beats one step below
299          //  printf("at %d = %d \n",v,beats[v]);
300        while (v != high){
301          beats[v]+= (PACE<<3);
302          //
303          if(beats[v] >=  3846){ //MAX_Y){
304             beats[v] = 0;
305             low++;
306             totalballs++;
307             if(scoretrigger ==1)
308             numballs = ((5*score)/totalballs+1);
309          }
310          WRITEBEAT(v);
311
312          v++;
313
314          if(v == NUMCELL)
315            v = 1;
316
317
318        }
319
320        //3) Check if boundaries are reached
321        if (high  == NUMCELL)          high = 1 ;
322        if (low   == NUMCELL)          low = 1 ;
323
324        //4) Reset the flags
```

```
        UP          = 0;
        NEWBEAT     = 0;


    }

  }
  return 0;
}
```

Listing 18: Scripts/conversion/beats.py

```
import sys
f = open(sys.argv[1])
a = f.readlines()[0].strip().split(" ")
print a
f.close()
```

Listing 19: Scripts/conversion/sprites.py

```
from PIL import Image
import sys
print "("
im = Image.open(sys.argv[1])
a = im.load()
for x in range(32):
  for y in range(32):
    if (a[x,y][3]>10):
      R,G,B,transp = a[x,y][0],a[x,y][1],a[x,y][2],0
    else:
      R,G,B,transp= 0,0,0,1
    outpixel = (transp << 30)+(R<<2)+(G<<12)+(B<<22)
    outpixel = hex(outpixel)
    padlength = (10-len(outpixel))
    outpixel = "x\""+"0"*padlength+outpixel[2:]+"\","
    if (y == 31):
        print outpixel[:-1]+"),\n("
    else:
        print outpixel
```

Listing 20: Scripts/conversion/toBinForFlashLight.rb

```
inf, out = File.open("new1.txt"), File.open("song.bin","wb")
  #Integer#pack('C') gets the binary representation (signed 8 bit int) of the number
  inf.read.split(" ").each {|sample| out.write([Integer(sample)].pack('c'))}
inf.close ; out.close
```

Listing 21: Scripts/matlab/wholescript.m

```
clear s;
clear sr;
clear array;
clear fp;
clear data;
```

```
fp=fopen('mew2.txt','w')
[s sr]=audioread('music12.wav');
s=s-min(s);
array=0:1/2^8*max(s):max(s)
for i=1:length(s)
    for j=1:length(array)-1
        if (s(i)>array(j) & s(i)<array(j+1))
            data(i)=j;
        end
    end
end
for i=1:length(data)
    fprintf(fp,'%d ',data(i));
end
clear s;
[s sr]=audioread('music12.wav');
b=beat(s,sr);
clear fp2;
fp2=fopen('beatsmew.txt','w');
for i=1:length(b)
    fprintf(fp2,'%d ',b(i));
end
```