Due Date: January 22, at 8:00 AM

CSE 4117 Final Project

In this project, you will use Bird-CPU to read keystrokes from a ps2 keyboard and draw a scene on a VGA monitor.

Your system will draw

- A spaceship bitmap
- and a planet bitmap
- on a green background.

Spaceship bitmap will be 16 pixels wide and 16 pixels high, and will be exactly at the center of the screen at the start up. Its design is given below (or, you can design your own). It will be red.

0000000010000000
0000000111000000
0000000111000000
0000000111000000
0000000111000000
0000001111100000
0000011111110000
0000111111111000
0001111111111100
0000000111000000
0000000111000000
0000000111000000
0000000111000000
0000001111100000
0000011111110000
0000000111000000

Planet bitmap is again 16x16, and it is of shape circle. Design one of your own. It is yellow. It will also start at the center of the VGA screen.

Motion of spaceship

The motions of spaceship will be controlled by the keyboard input:

--When the key W is pressed, the spaceship will move 4 pixels up.

--When the key S is pressed, the spaceship will move 4 pixels down.

--When the key A is pressed, the spaceship will move 4 pixels left.

--When the key D is pressed, the spaceship will move 4 pixels right.

-- spaceship should not pass the edges of the screen.

When the keys are released, no extra movement will occur (ie, you will disregard the second scan code which comes when you release the key).

Motion of the planet

The planet will move 10 pixels/second up in the vertical direction and 20 pixels/second to the right in the horizontal direction (these numbers are only approximate, do not try to make them exact). Whenever the planet hits left or right edge, its horizontal speed reverses. Whenever the planet hits top or bottom edge, its vertical speed reverses.

Hardware you will build

You can use the Verilog codes for the keyboard host controller interface and Mammal CPU without change from the website. The CPU will poll the keyboard for keypresses.

VGA module has the registers

Logic [15:0] x_spaceship, y_spaceship, x_planet, y_planet

Logic [15:0] spaceship_bitmap[0:15]

Logic [15:0] planet_bitmap[0:15]

VGA module will print the contents of the spaceship_bitmap and planet_bitmap to screen at every frame in such a way that the upper left hand corner of planet_bitmap will be at pixel (x_planet, y_planet) and the upper left hand corner of spaceship_bitmap will be at pixel (x_spaceship, y_spaceship). The algorithm will be

```
Infinite loop {
   For each (x,y)
     If pixel (x,y) is contained in spaceship bitmap
      --paint the pixel to the colour of the spaceship (red)
     elseif pixel (x,y) is contained in planet bitmap
       --paint the pixel to the colour of the planet (yellow)
      else
       --paint the pixel to the background color (green)
}
```

Note that the spaceship will always be at the front of the planet.

CPU will have two duties:

- During initialization, CPU will initialize spaceship_bitmap and planet_bitmap to the appropriate figures. This will require 16 store operations for each bitmap. Also don't forget that stack and IDT must be also initialized.
- During operation, CPU will change the registers x_spaceship, y_spaceship, x_planet, y_planet for each frame, hence cause the spaceship and the planet to move.

Communication between the Mammal CPU and the VGA module will be via interrupts. If CPU modifies the registers x_spaceship, y_spaceship, x_planet, y_planet while the cpu draws the

frame, artefacts could occur. Hence the modification of these registers should be done during the vertical flyback. When the vga module starts vertical flyback, it will send an interrupt to the CPU. CPU, on receiving the interrupt, will modify the registers in the ISR. During the modification, the VGA module will lower the interrupt signal. In all the other times, the CPU will poll the keyboard.

You could use the codes for the PS2 keyboard module and Mammal CPU without change (or, with very little change). You have to modify the top module which "glues" the CPU, keyboard, and monitor significantly. You are also required to modify the VGA module. Assign addresses to the keyboard and VGA registers.

On the software side, you have to write the assembly code which manages the hardware for the given task. Modify the assembler codes given in the lecture notes suitably, and fill in the blank parts to assemble your code.