Firmware Pacman-Qemu

Student: Mantvydas Luksas

Student ID: R00150390

On linux I have emulated the versatile application baseboard for ARM926EJ-S using QEMU. However, at the very start there is no firmware, so I had to use the firmware that was already created in the pacman-qemu.zip file. The firmware starts up with the help of the assembly language file ts.s, where the "reset_handler" invokes a boot loader which loads the operating system of the board. The reset_handler is part of the vector table of the system, where all the necessary processes memory locations are declared for the start up the system. The vector table is set up the ts.s file.

```
vectors_start:

  LDR PC, reset_handler_addr
  LDR PC, undef_handler_addr
  LDR PC, swi_handler_addr
  LDR PC, prefetch_abort_handler_addr
  LDR PC, data_abort_handler_addr
  B .
  LDR PC, irq_handler_addr
  LDR PC, fiq_handler_addr

reset_handler_addr:          .word reset_handler
undef_handler_addr:          .word undef_handler
swi_handler_addr:            .word swi_handler
prefetch_abort_handler_addr: .word prefetch_abort_handler
data_abort_handler_addr:     .word data_abort_handler
irq_handler_addr:            .word irq_handler
fiq_handler_addr:            .word fiq_handler

vectors_end:
```

The primary interrupt controller (PIC) controls all the interrupts so that the embedded system can run. The uart input peripheral is connected to this hardware which allows data to be displayed. From the primary interrupt controller IRQ is connected to CPU which states which device caused an interrupt. The IRQ handler is used in the t.c file where we determine which device caused the interrupt to occur. Two of the biggest interrupts are the timer and the uart0. The timer input peripheral is used for the sprites or the ghosts to move within the game board.

The uart peripheral stores all the keys entered by the user in the buffer. When we try to to extract a particular key from the buffer, we call the function upeek to get the key entered into the buffer. The upeek function is inside uart.c, which contains the uart handler which

determines which event occurred in the peripheral from the status register mis. Do_rx function is for received data for placing the typed keys into the buffer. And the do_tx function is for dumping data.

The vid.c is the video display controller for displaying the game, in other words all of the pixels that are part of the game. This is where the frame buffer is updated and changed, where the frame buffer shows a particular pixel 60 times per second. The vid.c primarily controls the screen.

The timer.c contains the timer_handler function where the time is determined in seconds based on the speed of the micro-controller. Inside of the timer the spriteMove variable is set in the timer_handler. So the speed can be set here for the ghosts.

The firmware also contains all the images that are needed to be used in the game indicated by the files ending in the .bmp extension. These images are 16 by 16 pixels as required for the table of our game.

The mk file contains all the necessary code to compile all the files including the images and in t.ld all the compiled images with the extension of .o are loaded which can then be accessed in the t.c file with extern char _binary_pacman_bmp_start for example.