Application Design

Kyle Lemmon

1 Firmware

Firmware package is preloaded into FPGA RAM. Firmware accesses file allocation table and finds the first and second entry. The length of the first entry is determined by the start address of the second address via software.

The firmware then loads the first file in the allocation table into memory starting at the first address. When the firmware has completed, the firmware jumps to the first memory address, handing control over to the kernel.

2 Kernel

The objective of the kernel is to provide a command-line interface for running programs that exist on the file system. The kernel provides all functions for interfacing with I/O that are called from within the kernel and from user programs executed on the processor.

3 User program

The user program is similar to any program run on a modern machine. The programs are written to the SD card and are loaded into memory by the firmware. They define their own functionality, but it is expected that for I/O from the SD card, the program uses the kernel's defined SD card access functions. The keyboard input and VGA output are handled by shared memory, which the user program may access itself.

4 Assembler considerations

The assembler must handle jumps to kernel code by outputting label spec files that contain the relative memory addresses of all labels in the compiled code. These files can be linked against in subsequent compilations to ensure that user programs can be linked against kernel subroutines.

4.1 Unified memory

We will be using a unified memory structure in order to load files into memory for execution. This allows us to easily include a .text section in assembly code. The assembler should support the .text section that will embed memory addresses for different types, including strings and integers.

4.2 Assembler Use Flow & Kernel Linking

The assembler must support linking accross files via a linking file. The flow for compiling and creating user programs is as follows:

1. Assemble Kernel → kernel.hex, kernel.ln

   kernel.ln is a text file that links a label with a memory address. This can be included in a user program assembly code, which allows the use of all labels from of kernel source assembly from within the user program code. It is similar to the a header file for the c standard library functions and memory addresses.

2. Assemble user program → prog.hex, prog.ln

   The assembler checks if any undefined labels used in the user program code are defined in the linking file, if so the address from the link file is used.

This system works without further hardware modification because the kernel is addressed by the firmware to start at address 0, so no offset is required for jumping to a kernel address. The link file also contains an end address, that is essentially the size in bytes of the compiled kernel code less one. This is used to calculate offsets for the user program's labels, since it exists above the kernel in memory.

This requires that all user programs are recompiled after a change to the kernel.

5 Memory Map

Addr, Len	Description
0x0000, kernel len	Kernel
kernel len, prog len	User program
...	Empty
0xE813, 1024	Kernel - user shared memory
0xEC13, 4800	VGA Glyph Memory
0xFED3, 300	Firmware - reset entry point!