OS Homework CSE506

111493601 KISHAN NERELLA
111481934 DINESH BADDAM

Homework

1. The existing boot loader maps every 1 GB of virtual address space to the lower 1 GB of physical address space. The task is to change the boot loader to map every 4 GB virtual address space to the lower 4GB physical address space.
2. Once the above is done, read and write to the ACHI controller without re-mapping its bar5

Work Summary

The most difficult part of the assignment was to find which files in the FreeBSD source code corresponding to the boot loader. In the existing code, each level 2 page table is pointing to 2 MB of 1 GB. This is done within 1 page. All the level 3 page tables entries point to this 'one' level 2 page. All the level 4 page tables entries point to the 'one' level 3 page table. We changed this to create 4 level-2 page tables each corresponding to [0GB-1GB), [1GB-2GB), [2GB-3GB) and [3GB-4GB). Every level-3 page table entry alternates between the above 'four' level 2 page tables. Every level 4 page table entry still points to the single level 3 page table. These new page tables are defined in the file amd64_tramp.S and the corresponding code is changed in elf64_freebsd.c. Along with this kernmem in sys/linker.script is changed to point to the last 4 GB of virtual address space instead of last 1 GB. To make the 2nd part of the homework work, just removed the bar remapping in sys/pci.c

Code files changes

In freebsd source code,

```
release/11.0.0/sys/boot/i386/libi386/amd64_tramp.S
release/11.0.0/sys/boot/i386/libi386/elf64_freebsd.c
```

In SBUNIX source code
`sys/linker.script`
`sys/pci.c`
`rootfs/boot/loader`

Building/Testing

We have submitted homework.tgz. Along with the usual files that are submitted for warmup projects, a new folder `homework/` is included which include elf64_freebsd.c, elf64_freebsd.c.old, amd64_tramp.S, amd64_tramp.S.old, loader.new and loader.old.

elf64_freebsd.c.old ---> Original freebsd source code
elf64_freebsd.c ---> Changed freebsd source code
amd64_tramp.S.old ---> Original freebsd source code
amd64_tramp.S ---> Changed freebsd source code

loader.old ---> Original loader that came with the handout
loader.new ---> Changed loader that we built

Once you extract the tar, do 'make' in the root directory. I already replaced `rootfs/boot/loader` with the updated boot loader.

Test 1 - Running the normal test using qemu is going to write to the 2nd SATA drive as mentioned in wp3.

Test 2 – This test is to ensure that the only difference before and after homework is just bar5 remapping and Test 1 is working only because of the loader changes. Replace `rootfs/boot/loader` with the original loader (can be obtained from `homework/loader.old`), '`make clean all`' and re-run the test. This should **FAIL**. Now uncomment line 53 in `sys/pci.c`, do a '`make clean all`' and re-run the test. This should **PASS**. The test passes now because we're doing the bar5 remapping again. You can comment back line 53 and replace `rootfs/boot/loader` with `homework/loader.new` to check that the new loader is working.

Building the loader-

If you want to build the loader from scratch, download freebsd 11.0.0 release source code. Replace `sys/boot/i386/libi386/amd64_tramp.S` and `sys/boot/i386/libi386/elf64_freebsd.c` with `homework/amd64_tramp.S` and `homework/elf64_freebsd.c` respectively. Do '`make`' inside '`sys/boot/i386/libi386`' directory and '`make loader`' in '`sys/boot/i386`'. You will get the new loader at '`sys/boot/i386/loader/loader`'. Replace `rootfs/boot/loader` with the newly built loader and do Test1 and Test2.