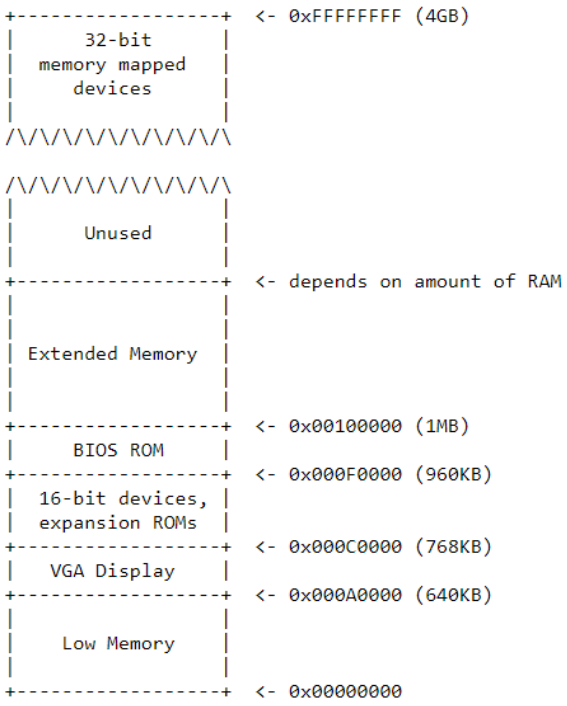
【6.828】Day2

Class	MIT 6.828
Date	@March 3, 2022

【Lecture 1】Lab 1

Part 1: PC Bootstrap

1.1 The PC's Physical Address Space



The first PC's, which were based on the 16-bit Intel 8088 processor, were only capable of addressing 1MB of physical memory. The physical address space of an early PC would therefore start at 0x0000000 but end at 0x000FFFFF.

The 640KB(0x000A0000) are marked "Low Memory" was the only random-access memory(RAM) that an early PC could use.

The 384KB area from 0x000A0000 through 0x000FFFFF was reserved by the hardware for special uses such as video display buffers and firmware held in non-volatile memory.

The most important part of this reserved area is the Basic Input/Output System(BIOS), which occupies the 64KB region from 0x000F0000 through 0x000FFFFF. The BIOS is responsible for performing basic system initialization such as activating the video card and checking the amount of memory installed.

After performing this initialization, the BIOS loads the operating system from some appropriate location such as floppy disk, hard disk, CD-ROM, or the network, and passes control of the machine to the operating system.

The PC architects nevertheless preserved the original layout for the low 1MB of physical address space in order to ensure backward compatibility with existing software.

More PCs therefore have a “hole” in physical memory from 0x000A0000 to 0x00100000, diving RAM into “low” or “conventional memory”(the first 640KB) and “extended memory”(everything else).

1.2 The ROM BIOS

After entering `make qemu-gdb`(which sets up QEMU and asks it to stop just before the processor execute the first instruction and waits for a debugging connection from GDB) and `make gdb`, we should see something like this:

```
athena% make gdb
GNU gdb (GDB) 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
+ target remote localhost:26000
The target architecture is assumed to be i8086
[f000:fff0] 0xffff0:    ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
+ symbol-file obj/kern/kernel
(gdb)
```

The following line:

```
[f000:fff0]0xffff0: ljmp $0xf000,$0xe05b
```

is GDB's disassembly of the first instruction to be executed. From this instruction, we can see:

1. The IBM PC starts executing at physical address 0x000ffff0, which is at the top of the 64KB area reserved for the ROM BIOS
2. The PC starts executing with `CS = 0xf000` and `IP = 0xfff0`
3. The first instruction to be executed is a `jmp` instruction, which jumps to the segmented address `CS = 0xf0000` and `IP = 0xe05b`

Why does QEMU start like this?

Because the BIOS in a PC is "hard-wired" to the physical address range 0x000f0000-0x000fffff, this design ensures that the BIOS always gets control of the machine first after power-up or any system restart - which is crucial because on power-up there is no other software anywhere in the machine's RAM that the processor could execute.

How does f000:fff0 get transferred into a physical address?

In real mode(the mode that PC starts off in), address translation works according to the formula: `physical address = 16 * segment + offset`, when the PC sets CS to 0xf000 and IP to 0xfff0, the physical address reference is

```
16 * 0xf000 + 0xfff0
= 0xf0000 + 0xfff0
= 0xffff0
```