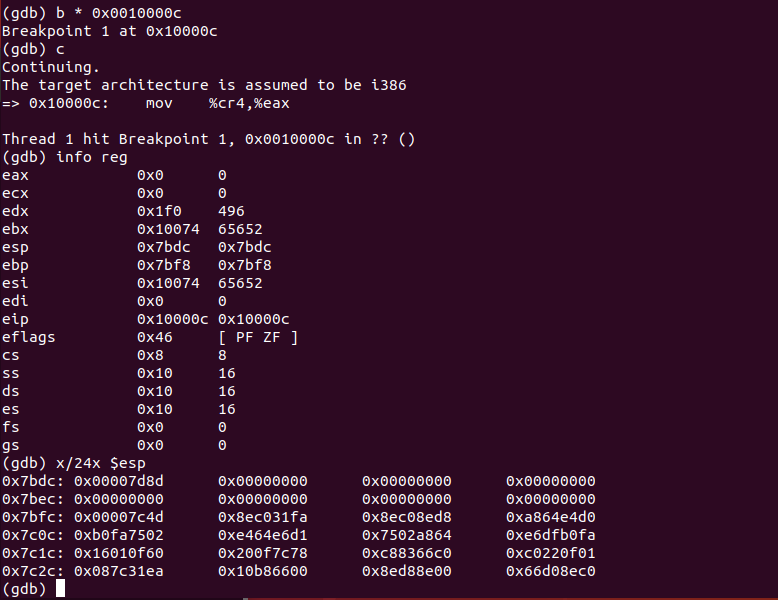
William Tran

2018/07/04

Homework (1) Report

My output of "info reg" and "x/24x $esp":



**Analysis:**

* Zero values on the stack: EAX, ECX, EDI, FS, and GS.
* List of non-zero value general registers:

EDX: 32-bit data register

EBX: 32-bit base register

ESP: 32-bit stack pointer

EBP: 32-bit stack base pointer

ESI: 32-bit source

EIP is an instruction pointer (it contains the address of the next instructor)

EFLAGS: 32-bit register that stores the results of operations and the state of the processor.

List of non-zero value segment registers:

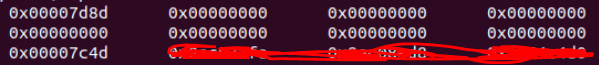
CS (code segment): pointer to the code

SS (stack segment): pointer to the stack

DS (data segment): pointer to the data

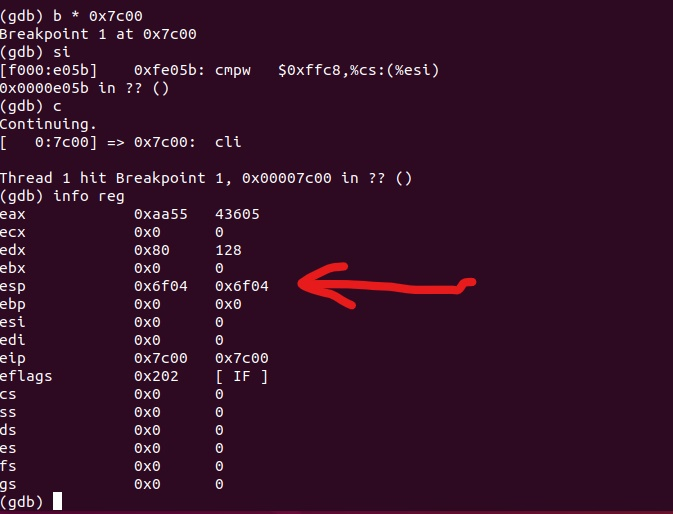
ES (extra segment): pointer to extra data (same for FS and GS if there is extra data)

In x/24x $esp, there are 24 segments of physical address corresponding to the address of the stack pointer. The stack pointer points to the top of the stack at 0x7bdc. The registers in the stack, therefore, are the zero values. The picture below shows the actual stack in the stack printout.

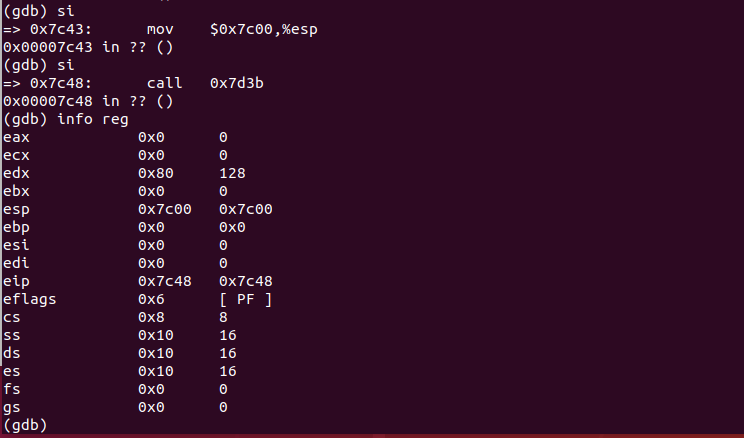


**Observation:**

After restarting qemu and gdb and setting a breakpoint at 0x7c00, which is the physical address that BIOS will load the 512-byte boot sector into, or the start of the boot block in bootasm.S, I noticed that esp, the stack pointer, has an address of 0x6f04, which is below 0x7c00. (see picture below)



I then single stepped until I saw a call, “call 0x7d3b”. This means bootmain is called. Also, ss, ds, and es changed from 0 to 16, indicating that the system is going into protected mode. I notice how the address of the stack pointer esp changed from 0x6f04 to 0x7c00, which means it has been initialized at the top of the stack. According to the file bootblock.asm, the stack pointer here is initialized right before entering bootmain.



The first instruction in bootmain here is *push %ebp,* which means it saves the frame pointer EBP to the stack. Then, it points $esp to the top of the stack again.



Finally, when the $eip changed to 0x0010000c, the program calls \*0x0010018, which means the entry address of the kernel. This address is then saved to the stack.