**Exercise 9.**Determine where the kernel initializes its stack, and exactly where in memory its stack is located. How does the kernel reserve space for its stack? And at which "end" of this reserved area is the stack pointer initialized to point to?

Looking at kernel.asm, at 0xf010002f, the move $0x0, %ebp clears the frame pointer register (which is ebp). The stack pointer is set by movl $(bootstacktop), %esp.

The next line initializes the stack pointer to f0100034, mov $0xf0110000, %esp. So, our stack is located at 0xf0110000.

**Exercise 10.** To become familiar with the C calling conventions on the x86, find the address of the test\_backtrace function in obj/kern/kernel.asm, set a breakpoint there, and examine what happens each time it gets called after the kernel starts. How many 32-bit words does each recursive nesting level of test\_backtrace push on the stack, and what are those words?

Set jumping point to address 0xf0100040, then continuing twice. We see that it adds 8 4 byte words because ebp value changes from 0xf010fff8 to 0xf010ffd8.

Note that, for this exercise to work properly, you should be using the patched version of QEMU available on the [tools](https://pdos.csail.mit.edu/6.828/2017/tools.html) page or on Athena. Otherwise, you'll have to manually translate all breakpoint and memory addresses to linear addresses. The JOS kernel should show the test\_backtrace running, as well as decimal 🡪 octal value.

**Exercise 11.** Implement the backtrace function as specified above. Use the same format as in the example, since otherwise the grading script will be confused. When you think you have it working right, run **make grade** to see if its output conforms to what our grading script expects, and fix it if it doesn't. *After* you have handed in your Lab 1 code, you are welcome to change the output format of the backtrace function any way you like.

If you use read\_ebp(), note that GCC may generate "optimized" code that calls read\_ebp() *before* mon\_backtrace()'s function prologue, which results in an incomplete stack trace (the stack frame of the most recent function call is missing). While we have tried to disable optimizations that cause this reordering, you may want to examine the assembly of mon\_backtrace() and make sure the call to read\_ebp() is happening after the function prologue.

In order to see changes in JOS kernel, you have to shut down JOS kernel and rerun.

