Se pide ahora escribir una nueva versión del archivo boot.S en que se defina el stack de arranque, así como el “entry point” \_start del kernel. Así, al saltar a código C, el stack ya estará debidamente configurado

// boot.S

#include "multiboot.h"

#define KSTACK\_SIZE 8192

.align 4

multiboot:

.long MULTIBOOT\_HEADER\_MAGIC

.long 0

.long -(MULTIBOOT\_HEADER\_MAGIC)

.globl \_start

\_start:

movl $0, %ebp

movl $kstack, %esp

push %ebp

mov $0, %ecx

testl $MULTIBOOT\_BOOTLOADER\_MAGIC, %eax

cmovne %ecx, %ebx

push %ebx

call kmain

halt:

hlt

jmp halt

.data

.p2align 12

kstack:

.space KSTACK\_SIZE

Finalmente: mostrar en una sesión de GDB los valores de %esp y %eip al entrar en kmain, así como los valores almacenados en el stack en ese momento.

El contenido del registro **%eax** es **0x2badb0ff**, boot.S coloca NULL en el stack.

$ make gdb

gdb -q -s kern2 -n -ex 'target remote 127.0.0.1:7508'

Reading symbols from kern2...done.

Remote debugging using 127.0.0.1:7508

warning: No executable has been specified and target does not support

determining executable automatically. Try using the "file" command.

0x0000fff0 in ?? ()

(gdb) b kmain

Breakpoint 1 at 0x100064: file kern2.c, line 5.

(gdb) c

Continuing.

Breakpoint 1, kmain (mbi=0x9500) at kern2.c:5

5 void kmain(const multiboot\_info\_t \*mbi) {

(gdb) p/x $eip

$1 = 0x100064

(gdb) p/x $esp

$2 = 0x101ff4

(gdb) x/4xw $esp

0x101ff4: 0x0010001d 0x00009500 0x00000000 0x00000000