```
**********************************************************
   25287 Sat Feb  7 18:57:37 2015
new/usr/src/uts/armv6/os/fakebop.c
armv6: bop_panic should hexdump the stack
It's the little things that make debugging easier.
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2014 Joyent, Inc.  All rights reserved.
 * Copyright (c) 2015 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */

/*
 * Just like in i86pc, we too get the joys of mimicking the SPARC boot system.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/bootconf.h>
#include <sys/bootsvcs.h>
#include <sys/boot_console.h>
#include <sys/atag.h>
#include <sys/varargs.h>
#include <sys/cmn_err.h>
#include <sys/sysmacros.h>
#include <sys/systm.h>
#include <sys/ctype.h>
#include <sys/bootstat.h>
#include <sys/privregs.h>
#include <sys/cpu_asm.h>
#include <sys/boot_mmu.h>
#include <sys/elf.h>
#include <sys/archsystm.h>
#endif /* ! codereview */

static bootops_t bootop;

/*
 * Debugging help
 */
static int fakebop_prop_debug = 0;
static int fakebop_alloc_debug = 0;
static int fakebop_atag_debug = 0;

static uint_t kbm_debug = 1;
#define DBG_MSG(x)      { if (kbm_debug) bcons_puts(x); bcons_puts("\n"); }
#define BUFFERSIZE      256
static char buffer[BUFFERSIZE];

/*
 * fakebop memory allocations scheme
 *
 * It's a virtual world out there. The loader thankfully tells us all the areas
 * that it has mapped for us and it also tells us about the page table arena --
 * a set of addresses that have already been set aside for us. We have two
 * different kinds of allocations to worry about:
```



```
 *
 *    o Those that specify a particular vaddr
 *    o Those that do not specify a particular vaddr
 *
 * Those that do not specify a particular vaddr will come out of our scratch
 * space which is a fixed size arena of 16 MB (FAKEBOP_ALLOC_SIZE) that we set
 * aside at the beginning of the allocator. If we end up running out of that
 * then we'll go ahead and figure out a slightly larger area to worry about.
 *
 * Now, for those that do specify a particular vaddr we'll allocate more
 * physical address space for it. The loader set aside enough L2 page tables for
 * us that we'll go ahead and use the next 4k aligned address.
 */
#define FAKEBOP_ALLOC_SIZE      (16 * 1024 * 1024)

static size_t bop_alloc_scratch_size;
static uintptr_t bop_alloc_scratch_next;        /* Next scratch address */
static uintptr_t bop_alloc_scratch_last;        /* Last scratch address */

static uintptr_t bop_alloc_pnext;               /* Next paddr */
static uintptr_t bop_alloc_plast;               /* cross this paddr and panic */

#define BI_HAS_RAMDISK  0x1

/*
 * TODO Generalize this
 * This is the set of information tha we want to gather from the various atag
 * headers. This is simple and naive and will need to evolve as we have
 * additional boards beyond just the RPi.
 */
typedef struct bootinfo {
        uint_t          bi_flags;
        char            *bi_cmdline;
        uint32_t        bi_ramdisk;
        uint32_t        bi_ramsize;
} bootinfo_t;

static bootinfo_t bootinfo;     /* Simple set of boot information */

static struct boot_syscalls bop_sysp = {
        bcons_getchar,
        bcons_putchar,
        bcons_ischar,
};

/*
 * stuff to store/report/manipulate boot property settings.
 */
typedef struct bootprop {
        struct bootprop *bp_next;
        char *bp_name;
        uint_t bp_vlen;
        char *bp_value;
} bootprop_t;

static bootprop_t *bprops = NULL;

static void
hexdump_stack()
{
        extern char t0stack[];

        uint8_t *start = (uint8_t *)t0stack;
        uint8_t *end = start + DEFAULTSTKSZ;
        uint8_t *ptr;
        int i;
```

```
128         bop_printf(NULL, "stack (fp = %x):\n", getfp());

130         ptr = (uint8_t *)(getfp() & ~0xf);
131         if (ptr <= start || ptr >= end)
132                 ptr = start;

134         while (ptr < end) {
135                 uint32_t *tmp = (uint32_t *)ptr;

137                 bop_printf(NULL, "%p: %08x %08x %08x %08x\n", ptr,
138                     tmp[0], tmp[1], tmp[2], tmp[3]);

140                 ptr += 16;
141         }

143 }

145 #endif /* ! codereview */
146 void
147 bop_panic(const char *msg)
148 {
149         bop_printf(NULL, "ARM bop_panic:\n%s\n", msg);

151         hexdump_stack();

153         bop_printf(NULL, "Spinning Forever...", msg);
 37         bop_printf(NULL, "ARM bop_panic:\n%s\nSpinning Forever...", msg);
154         for (;;)
155                 ;
156 }
```

*unchanged_portion_omitted_*