```
/*
 * (C) Copyright 2002
 * Sysgo Real-Time Solutions, GmbH <www.elinos.com>
 * Marius Groeger <mgroeger@sysgo.de>
 *
 * Copyright (C) 2001  Erik Mouw (J.A.K.Mouw@its.tudelft.nl)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 */

#include <common.h>
#include <command.h>
#include <cmd_boot.h>
#include <image.h>
#include <zlib.h>
#include <asm/byteorder.h>
#ifdef CONFIG_HAS_DATAFLASH
#include <dataflash.h>
#endif

#include <asm/setup.h>
#define tag_size(type)  ((sizeof(struct tag_header) + sizeof(struct type)) >> 2)
#define tag_next(t)     ((struct tag *)((u32 *)(t) + (t)->hdr.size))

#if defined (CONFIG_SETUP_MEMORY_TAGS) || \
    defined (CONFIG_CMDLINE_TAG) || \
    defined (CONFIG_INITRD_TAG) || \
    defined (CONFIG_VFD)
static void setup_start_tag(bd_t *bd);
# ifdef CONFIG_SETUP_MEMORY_TAGS
static void setup_memory_tags(bd_t *bd);
# endif
static void setup_commandline_tag(bd_t *bd, char *commandline);
#if 0
static void setup_ramdisk_tag(bd_t *bd);
#endif
# ifdef CONFIG_INITRD_TAG
static void setup_initrd_tag(bd_t *bd, ulong initrd_start, ulong initrd_end);
# endif
static void setup_end_tag(bd_t *bd);
# if defined (CONFIG_VFD)
static void setup_videolfb_tag(gd_t *gd);
# endif


static struct tag *params;
#endif /* CONFIG_SETUP_MEMORY_TAGS || CONFIG_CMDLINE_TAG || CONFIG_INITRD_TAG */

#ifdef CONFIG_SHOW_BOOT_PROGRESS
# include <status_led.h>
# define SHOW_BOOT_PROGRESS(arg)        show_boot_progress(arg)
#else
# define SHOW_BOOT_PROGRESS(arg)
#endif

extern image_header_t header;           /* from cmd_bootm.c */

/* ghcstop */
/*
 * armboot용(즉, arm cpu용) bootm 명령에서 linux를 부팅하는 명령
 * ppc같은 경우는 common/cmd_bootm.c 에서 이루어 지지만 arm일 경우는 extern을
 * 선언해 놓고 여기로 jump를 한다.
 * ppc linux일 경우는 bd_t등을 커널로 건네주지만 여기서는 그런걸 건네주지 않고
```

```
 * 다만 architecture id등만 건네주게 되어 있다. 그래서리...함수를 틀리게 쓴것 같다
 * ppc linux일 경우와 마찬가지로 부트로더에서 커널 압축을
 * 해제하는 것 같다.... 기존의 arm커널은 그렇게 되어 있지 않으나, u-boot를 사용할
 * 때는 부트로더에서 풀어주기 때문에 코드를 틀리게 쓴 것 같음...
 * 자세한 자료는 armboot 포팅문서와 u-boot & linux manual과 denx의 linuxarm커널을
 * 참조할 것
 * ※ 스타일이 blob이랑 비슷하네요...^^
 *
 * 이 예제는 로딩 번지를 바꿔줘야 한다...특히 arm일때는 하여간, 여기서는 이렇게
 * 한다는 것만 설명한다.(linuxarm의 arch/arm/boot/Makefile 참조...)
 * - 커널만 로딩할 경우(ppc405예제다...2410의 경우는 33000000으로 로딩...)
 * ex> bootm 100000
 * - ramdisk image도 로딩했을 경우
 * ex> bootm 100000 200000
 */
void do_bootm_linux(cmd_tbl_t *cmdtp, int flag, int argc, char *argv[],
                ulong addr, ulong *len_ptr, int   verify)
{
    DECLARE_GLOBAL_DATA_PTR;

    ulong len = 0, checksum;
    ulong initrd_start, initrd_end;
    ulong data;
    void (*theKernel)(int zero, int arch);
    image_header_t *hdr = &header;
    bd_t *bd = gd->bd;
#ifdef CONFIG_CMDLINE_TAG
    char *commandline = getenv("bootargs");
#endif

    /* 커널 이미지 header에 있는 entry point address를 host byte순서로 바꿔서
    theKernel이란 함수의 형태로 설정한다....^^
    */
    theKernel = (void (*)(int, int))ntohl(hdr->ih_ep);

    /*
     * Check if there is an initrd image
     * : argument가 세개 이상일 경우 ramdisk image도 같이 있는 경우이다.
     */
    if (argc >= 3) {
        SHOW_BOOT_PROGRESS (9);

        /* 세번째 argument의 주소를 받아온다...: ramdisk image가 loading되어 있는 주소*/
        addr = simple_strtoul(argv[2], NULL, 16);

        printf ("## Loading Ramdisk Image at %08lx ...\n", addr);

        /* Copy header so we can blank CRC field for re-calculation */
#ifdef CONFIG_HAS_DATAFLASH
        if (addr_dataflash(addr)){
                read_dataflash(addr, sizeof(image_header_t), (char *)&header);
        } else
#endif
        /* cmd_bootm.c에 선언된 global변수인 header로 ramdisk image header를 가지고 온다
        여기서는 커널 헤더에 대한 것이 필요없으므로...재활용하는거 같다...^^ */
        memcpy (&header, (char *)addr, sizeof(image_header_t));

        /* magic number가 틀릴경우...reset */
        if (ntohl(hdr->ih_magic) != IH_MAGIC) {
            printf ("Bad Magic Number\n");
            SHOW_BOOT_PROGRESS (-10);
            do_reset (cmdtp, flag, argc, argv); /* ==> cpu/arm920t/cpu.c */
        }

        /* ramdisk image header에 대한 checksum 계산, 틀렸을 경우 reset */
        data = (ulong)&header;
        len  = sizeof(image_header_t);

        checksum = ntohl(hdr->ih_hcrc);
        hdr->ih_hcrc = 0;

        if (crc32 (0, (char *)data, len) != checksum) {
            printf ("Bad Header Checksum\n");
            SHOW_BOOT_PROGRESS (-11);
            do_reset (cmdtp, flag, argc, argv);/* ==> cpu/arm920t/cpu.c */
        }
```

```
        SHOW_BOOT_PROGRESS (10);

        /* header image에 대한 정보 print: ==> common/cmd_bootm.c */
        print_image_hdr (hdr);

        /* header를 제외한 실제 ramdisk image의 시작주소와 길이를 설정 */
        data = addr + sizeof(image_header_t);
        len  = ntohl(hdr->ih_size);

#ifdef CONFIG_HAS_DATAFLASH
        if (addr_dataflash(addr)){
                read_dataflash(data, len, (char *)CFG_LOAD_ADDR);
                data = CFG_LOAD_ADDR;
        }
#endif

        /* verify란 환경변수를 읽어와서 1이냐 0으로냐 세팅이 됨, do_bootm()에서
        이게 1로 되어 있을 경우 ramdisk image에 대한 checksum을 계산한다
        틀리면 reset */
        if (verify) {
            ulong csum = 0;

            printf ("   Verifying Checksum ... ");
            csum = crc32 (0, (char *)data, len);
            if (csum != ntohl(hdr->ih_dcrc)) {
                printf ("Bad Data CRC\n");
                SHOW_BOOT_PROGRESS (-12);
                do_reset (cmdtp, flag, argc, argv);
            }
            printf ("OK\n");
        }

        SHOW_BOOT_PROGRESS (11);

        /* ramdisk image의 헤더를 조사해서 이미지 타입이 틀릴 경우 reset */
        if ((hdr->ih_os   != IH_OS_LINUX)       ||
            (hdr->ih_arch != IH_CPU_ARM)        ||
            (hdr->ih_type != IH_TYPE_RAMDISK)   ) {
            printf ("No Linux ARM Ramdisk Image\n");
            SHOW_BOOT_PROGRESS (-13);
            do_reset (cmdtp, flag, argc, argv);
        }

        /*
         * Now check if we have a multifile image
         : argument가 두개인데 커널만 있는게 아니라 커널이랑 ramdisk등이 합쳐진
           이미지일 경우
         */
    } else if ((hdr->ih_type==IH_TYPE_MULTI) && (len_ptr[1])) {
        ulong tail    = ntohl(len_ptr[0]) % 4;
        int i;

        SHOW_BOOT_PROGRESS (13);

        /* skip kernel length and terminator */
        data = (ulong)(&len_ptr[2]);
        /* skip any additional image length fields */
        for (i=1; len_ptr[i]; ++i)
          data += 4;
        /* add kernel length, and align */
        data += ntohl(len_ptr[0]);
        if (tail) {
            data += 4 - tail;
        }

        len   = ntohl(len_ptr[1]);

    } else {
        /*
         * no initrd image
         */
        SHOW_BOOT_PROGRESS (14);

        data = 0;
    }
```

```
#ifdef  DEBUG
    if (!data) {
        printf ("No initrd\n");
    }
#endif

    /* ramdisk image가 있을 경우 ramdisk image의 시작포인트랑 끝 포인트를 지정한다 */
    if (data) {
        initrd_start = data;
        initrd_end   = initrd_start + len;
    } else { /* 없을 경우는 그냥 0으로 지정...*/
        initrd_start = 0;
        initrd_end = 0;
    }

    SHOW_BOOT_PROGRESS (15);

#ifdef DEBUG
    printf ("## Transferring control to Linux (at address %08lx) ...\n",
            (ulong)theKernel);
#endif

/*
(ghcstop_caution)
이쪽 코드는 kelp의 권수호님이 설명해 놓은 blob설명서를 보면 잘 나와 있는 것 같다.

커널에서 쓰는 initrd등의 번지를 건네주는게 여기 밖에 없다. ppc와는 틀리다.

여기서 parameter가 들어가는 번지와 커널에서 부트 파라미터(tag)를 받는 번지를
서로 맞춰줘야 한다.

setup_start_tag() 에서
params = (struct tag *)bd->bi_boot_params;
를 설정하는데 이 bi_boot_params영역을 커널에서 받는 번지와 맞춰줘야 될 거 같다.

board/smdk2410/smdk2410.c의 board_init()에서 gd->bd->bi_boot_params = 0x30000100;
로 설정이 된다. 이 부분은 커널과 맞춰줘야 한다.

※ 커널에서는 이 부분과 맞춰주면 된다.

arch/arm/kernel/setup.c의 setup_arch() 함수에서 parse_tags():
부트로더에서 넣어준 tag가 들어간 메모리상의(실제 물리메모리의 주소겠지)
tag들을 파싱해서 각종 변수들을 설정한다.

부트로더와 s3c2410에 대한 architecture 설정 스트럭쳐 :
mizi는 arch/arm/mach-s3c2410/smdk.c 에 선언
swl-patch는 arch/arm/mach-s3c2410/arch.c 에 선언
된 MCAHINE_START 매크로와 MACHINE_END 매크로 사이에 들어가는 BOOT_PARAMS()에
부트로더의 tag가 들어가는 번지와 설정을 맞출 것.
U-BOOT일 경우

board/smdk2410/smdk2410.c의 board_init()에서 gd->bd->bi_boot_params = 0x30000100;

로 설정이 된다. 이 부분과 lib_arm/armlinux.c의 do_bootm_linux()의
setup_start_tag()부분을 볼 것

*/
#if defined (CONFIG_SETUP_MEMORY_TAGS) || \
    defined (CONFIG_CMDLINE_TAG) || \
    defined (CONFIG_INITRD_TAG) || \
    defined (CONFIG_VFD)
    setup_start_tag(bd); /* bd의 parameter영역을 params란데로 대입해서
                            여러 parameter들을 bd로 세팅이 되게 만든다 */
#ifdef CONFIG_SETUP_MEMORY_TAGS
    setup_memory_tags(bd);
#endif
#ifdef CONFIG_CMDLINE_TAG
    setup_commandline_tag(bd, commandline);
#endif
#ifdef CONFIG_INITRD_TAG
    setup_initrd_tag(bd, initrd_start, initrd_end);
#endif
#if 0
    setup_ramdisk_tag(bd);
#endif
```

```
#if defined (CONFIG_VFD)
    setup_videolfb_tag(gd);
#endif
    setup_end_tag(bd);
#endif

    /* we assume that the kernel is in place */
    printf("\nStarting kernel ...\n\n");

    /* ==> cpu/arm920t/cpu.c: I/D cache turn-off & I/D cache flush */
    cleanup_before_linux();

    /* 커널 이미지로 제어권을 넘겨준다.

    여기서 넘겨진 argument는 r0에 0이 들어가고 r1에는 architecture number가 들어간다. */
    theKernel(0, bd->bi_arch_number);
}


#if defined (CONFIG_SETUP_MEMORY_TAGS) || \
    defined (CONFIG_CMDLINE_TAG) || \
    defined (CONFIG_INITRD_TAG) || \
    defined (CONFIG_VFD)
static void setup_start_tag(bd_t *bd)
{

        /*
         * board/smdk2410/smdk2410.c의 board_init()에서
         * gd->bd->bi_boot_params = 0x30000100;
         * 로 설정되었음. 이건 커널에서 받는 넘이랑 맞춰줄 것
         */
    params = (struct tag *)bd->bi_boot_params;

    params->hdr.tag = ATAG_CORE;
    params->hdr.size = tag_size(tag_core);

    params->u.core.flags = 0;
    params->u.core.pagesize = 0;
    params->u.core.rootdev = 0;

    params = tag_next(params);
}


#ifdef CONFIG_SETUP_MEMORY_TAGS
static void setup_memory_tags(bd_t *bd)
{
    int i;

    for(i = 0; i < CONFIG_NR_DRAM_BANKS; i++) {
        params->hdr.tag = ATAG_MEM;
        params->hdr.size = tag_size(tag_mem32);

        params->u.mem.start = bd->bi_dram[i].start;
        params->u.mem.size = bd->bi_dram[i].size;

        params = tag_next(params);
    }
}
#endif  /* CONFIG_SETUP_MEMORY_TAGS */


static void setup_commandline_tag(bd_t *bd, char *commandline)
{
    char *p;

    /* eat leading white space */
    for(p = commandline; *p == ' '; p++)
      ;

    /* skip non-existent command lines so the kernel will still
     * use its default command line.
     */
    if(*p == '\0')
      return;
```

```
    params->hdr.tag = ATAG_CMDLINE;
    params->hdr.size = (sizeof(struct tag_header) + strlen(p) + 1 + 4) >> 2;

    strcpy(params->u.cmdline.cmdline, p);

    params = tag_next(params);
}


#ifndef ATAG_INITRD2
#define ATAG_INITRD2    0x54420005
#endif

#ifdef CONFIG_INITRD_TAG
static void setup_initrd_tag(bd_t *bd, ulong initrd_start, ulong initrd_end)
{
    /* an ATAG_INITRD node tells the kernel where the compressed
     * ramdisk can be found. ATAG_RDIMG is a better name, actually.
     */
    params->hdr.tag = ATAG_INITRD2;
    params->hdr.size = tag_size(tag_initrd);

    params->u.initrd.start = initrd_start;
    params->u.initrd.size = initrd_end - initrd_start;

    params = tag_next(params);
}
#endif  /* CONFIG_INITRD_TAG */


#if 0
static void setup_ramdisk_tag(bd_t *bd)
{
    /* an ATAG_RAMDISK node tells the kernel how large the
     * decompressed ramdisk will become.
     */
    params->hdr.tag = ATAG_RAMDISK;
    params->hdr.size = tag_size(tag_ramdisk);

    params->u.ramdisk.start = 0;
    /*params->u.ramdisk.size = RAMDISK_SIZE; */
    params->u.ramdisk.flags = 1;        /* automatically load ramdisk */

    params = tag_next(params);
}
#endif /* 0 */

#if defined (CONFIG_VFD)
static void setup_videolfb_tag(gd_t *gd)
{
    /* An ATAG_VIDEOLFB node tells the kernel where and how large
     * the framebuffer for video was allocated (among other things).
     * Note that a _physical_ address is passed !
     *
     * We only use it to pass the address and size, the other entries
     * in the tag_videolfb are not of interest.
     */
    params->hdr.tag = ATAG_VIDEOLFB;
    params->hdr.size = tag_size(tag_videolfb);

    params->u.videolfb.lfb_base = (u32)gd->fb_base;
    /* 7168 = 256*4*56/8 - actually 2 pages (8192 bytes) are allocated */
    params->u.videolfb.lfb_size = 7168;

    params = tag_next(params);
}
#endif

static void setup_end_tag(bd_t *bd)
{
    params->hdr.tag = ATAG_NONE;
    params->hdr.size = 0;
}

#endif /* CONFIG_SETUP_MEMORY_TAGS || CONFIG_CMDLINE_TAG || CONFIG_INITRD_TAG */
```