망고100 보드로 놀아보자-8

U-boot 실행 Sequence

cpu/s5pc1xx/start.s – Jump vector table

```
/*
 *************************************************************************
 *
 * Jump vector table as in table 3.1 in [1]
 *
 *************************************************************************
 */


.globl _start
_start:     b       reset
        ldr     pc, _undefined_instruction
        ldr     pc, _software_interrupt
        ldr     pc, _prefetch_abort
        ldr     pc, _data_abort
        ldr     pc, _not_used
        ldr     pc, _irq
        ldr     pc, _fiq

_undefined_instruction:   .word undefined_instruction
_software_interrupt:      .word software_interrupt
_prefetch_abort:      .word prefetch_abort
_data_abort:          .word data_abort
_not_used:            .word not_used
_irq:               .word irq
_fiq:               .word fiq

     .balignl 16,0xdeadbeef
```

ARM 이 exception 이 걸리면 각 예외에 따라 무조건 정해진 해당번지로 jump

Exception vector의 위치를 word(4byte)로 정의

cpu/s5pc1xx/start.s – startup code (1)

```
_TEXT_BASE:
     .word     TEXT_BASE

.globl _armboot_start
_armboot_start:
     .word _start

/*
 * These are defined in the board-specific linker script.
 */
.globl _bss_start
_bss_start:
     .word __bss_start

.globl _bss_end
_bss_end:
     .word _end

#ifdef CONFIG_USE_IRQ
/* IRQ stack memory (calculated at run-time) */
.globl IRQ_STACK_START
IRQ_STACK_START:
     .word     0x0badc0de

/* IRQ stack memory (calculated at run-time) */
.globl FIQ_STACK_START
FIQ_STACK_START:
     .word 0x0badc0de
#endif
```

각 보드에 대한 config.mk 는 board/XXXXX/config.mk 에 선언되어 있다. 여기서는 TEXT_BASE = 0x33F80000

_start 라는 위치로 __armboot_start 을 설정

cpu/s5pc1xx/start.s – SVC32 mode

```
reset:
        /*
         * set the cpu to SVC32 mode and IRQ & FIQ disable
         */
        mrs     r0,cpsr
        bic     r0,r0,#0x1f
        orr     r0,r0,#0xd3
        msr     cpsr,r0
```

CPSR 을 R0로 읽어온다.

R0 := R0 & 0xFFFFFFE0 (하위 5 비트의 mode 비트를 clear 함)

R0 := R0 | 0x11010011 (supervisor 모드로 변경)

R0 의 값을 CPSR로 로딩

cpu/s5pc1xxx/start.s – cpu_init_crit

CP15 : system control processor
-cache, MMU, protection system, clocking mode, big /litter endian operation 과 같은 arm920t 의 다른 시스템 옵션들을 설정하고 제어함.
- MCR, MRC 명령으로 접근 가능

```
cpu_init_crit:
        /*
         * Invalidate L1 I/D
         */
        mov     r0, #0                  @ set up for MCR
        mcr     p15, 0, r0, c8, c7, 0   @ invalidate TLBs
        mcr     p15, 0, r0, c7, c5, 0   @ invalidate icache

        /*
         * disable MMU stuff and caches
         */
        mrc     p15, 0, r0, c1, c0, 0
        bic     r0, r0, #0x00002000     @ clear bits 13 (--V-)
        bic     r0, r0, #0x00000007     @ clear bits 2:0 (-CAM)
        orr     r0, r0, #0x00000002     @ set bit 1 (--A-) Align
        orr     r0, r0, #0x00000800     @ set bit 12 (Z---) BTB
        mcr     p15, 0, r0, c1, c0, 0
```

4.2 OPERATING MODE REGISTER (OMR, R, ADDRESS = 0XE000_0004)

Field	Bit	Description
Reserved	[31:18]	Read as zero
Reserved	[17]	Read as one
Reserved	[16]	Read as one
Reserved	[15:14]	Read as zero
NFMOD	[13:8]	The value of the XNFMOD[5:0] pad
Reserved	[7:5]	Read as zero
OM	[4:0]	The value of the XOM[4:0] pad

```
/* Read booting information */
ldr     r0, =PRO_ID_BASE
ldr     r1, [r0,#OMR_OFFSET]
bic     r2, r1, #0xfffffff9
cmp     r2, #0x0
moveq   r3, #BOOT_NAND
cmp     r2, #0x2
moveq   r3, #BOOT_ONENAND
cmp      r2, #0x4
moveq    r3, #BOOT_MMCSD

ldr     r0, =INF_REG_BASE
str     r3, [r0, #INF_REG3_OFFSET]
```

OM[0:4] Pin 정보를 읽어서 information REG 에 저장

cpu/s5pc1xxx/start.s – Boot mode

```
        ldr     r0, =0xff000fff
        bic     r1, pc, r0              /* r
        ldr     r2, _TEXT_BASE          /* r
        bic     r2, r2, r0              /* r
        cmp     r1, r2                  /* c
        beq     after_copy              /* r

        ldr     r0, =INF_REG_BASE
        ldr     r1, [r0, #INF_REG3_OFFSET]
        cmp     r1, #BOOT_NAND          /* C
        beq     nand_boot
        cmp     r1, #BOOT_ONENAND       /* C
        beq     onenand_boot
        cmp     r1, #BOOT_MMCSD
        beq     mmcsd_boot

nand_boot:
        mov     r0, #0x1000
        bl      copy_from_nand
        b       after_copy
```

Info regster에서 정보를 읽어서 Nand boot, mmcsd_boot로 분기

Lowlevel_init ?

```
bl      lowlevel_init   /* go setup pll,mux,memory */
```

Cpu/s5pc1xx/start.s에서 call

```
lowlevel_init:

        ldr     sp, =0x37ff0    /* setup temp stack pointer */
        sub     sp, sp, #12
        push {lr}

        /* IO Retention release */
        ldr     r0, =(ELFIN_CLOCK_POWER_BASE + OTHERS_OFFSET)
        ldr     r1, [r0]
        ldr     r2, =IO_RET_REL
        orr     r1, r1, r2
        str     r1, [r0]

        /* Disable Watchdog */
        ldr     r0, =0xEA200000
        mov     r1, #0
        str     r1, [r0]

        /* CS0 - 16bit SRAM, Enable nBE */
        ldr     r0, =ELFIN_SROM_BASE
        mov     r1, #0x9
        str     r1, [r0]


        /* init system clock */
        bl system_clock_init
        /* for UART */
        bl uart_asm_init

        bl dma_init
#if defined(CONFIG_NAND)
        /* NAND GPIO PIN MUX */
        bl nand_pin_mux

        /* simple init for NAND */
        bl nand_asm_init
#endif
```

Watchdog,Clock, UART, Nand 을 초기화

cpu/s5pc1xx/start.s – setup the stack

```
stack_setup:
#if defined(CONFIG_MEMORY_UPPER_CODE)
        ldr     sp, =(CFG_UBOOT_BASE + CFG_UBOOT_SIZE - 0x1000)
#else
        ldr     r0, _TEXT_BASE          /* upper 128 KiB: relocated uboot
        sub     r0, r0, #CFG_MALLOC_LEN /* malloc area
        sub     r0, r0, #CFG_GBL_DATA_SIZE /* bdinfo
#if defined(CONFIG_USE_IRQ)
        sub     r0, r0, #(CONFIG_STACKSIZE_IRQ+CONFIG_STACKSIZE_FIQ)
#endif
        sub     sp, r0, #12             /* leave 3 words for abort-stack
```

Stack address 를 지정하였으므로 드디어 RAM에서 코드가 동작할 수 있는 조건이 마련되었다.

Stack pointer의 최상위 3 개 word를 비워두는 것은 abort exception 발생하면, exception 발생하기 직전 PC 와 CPRS를 저장하여 debugging 정보로 이용하기 위해서이다.

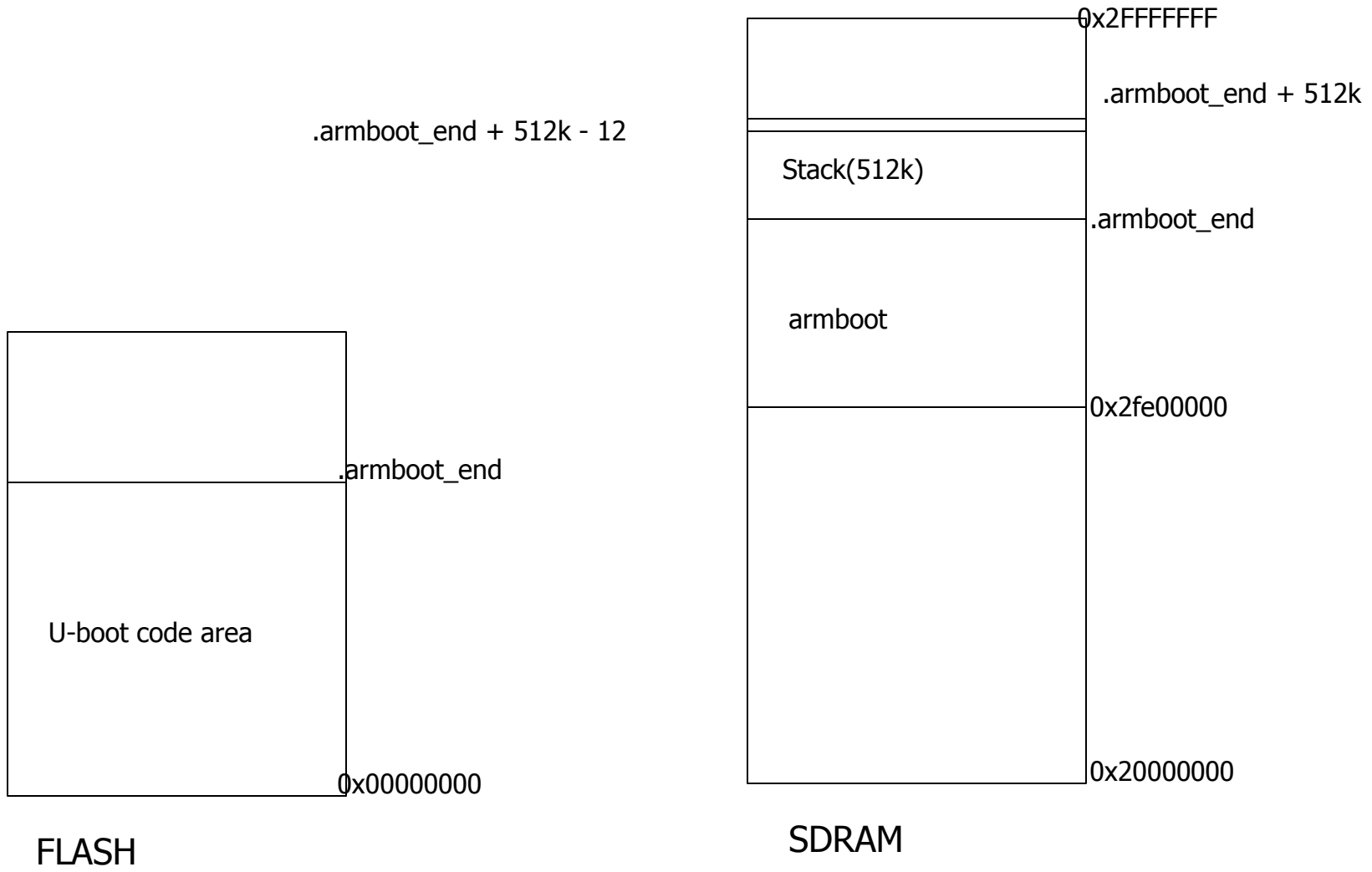
```
clear_bss:
        ldr     r0, _bss_start
        ldr     r1, _bss_end
        mov     r2, #0x00000000

clbss_l:
        str     r2, [r0]
        add     r0, r0, #4
        cmp     r0, r1
        ble     clbss_l

        ldr     pc, _start_armboot
```

Global 변수 들 0 으로 초기화

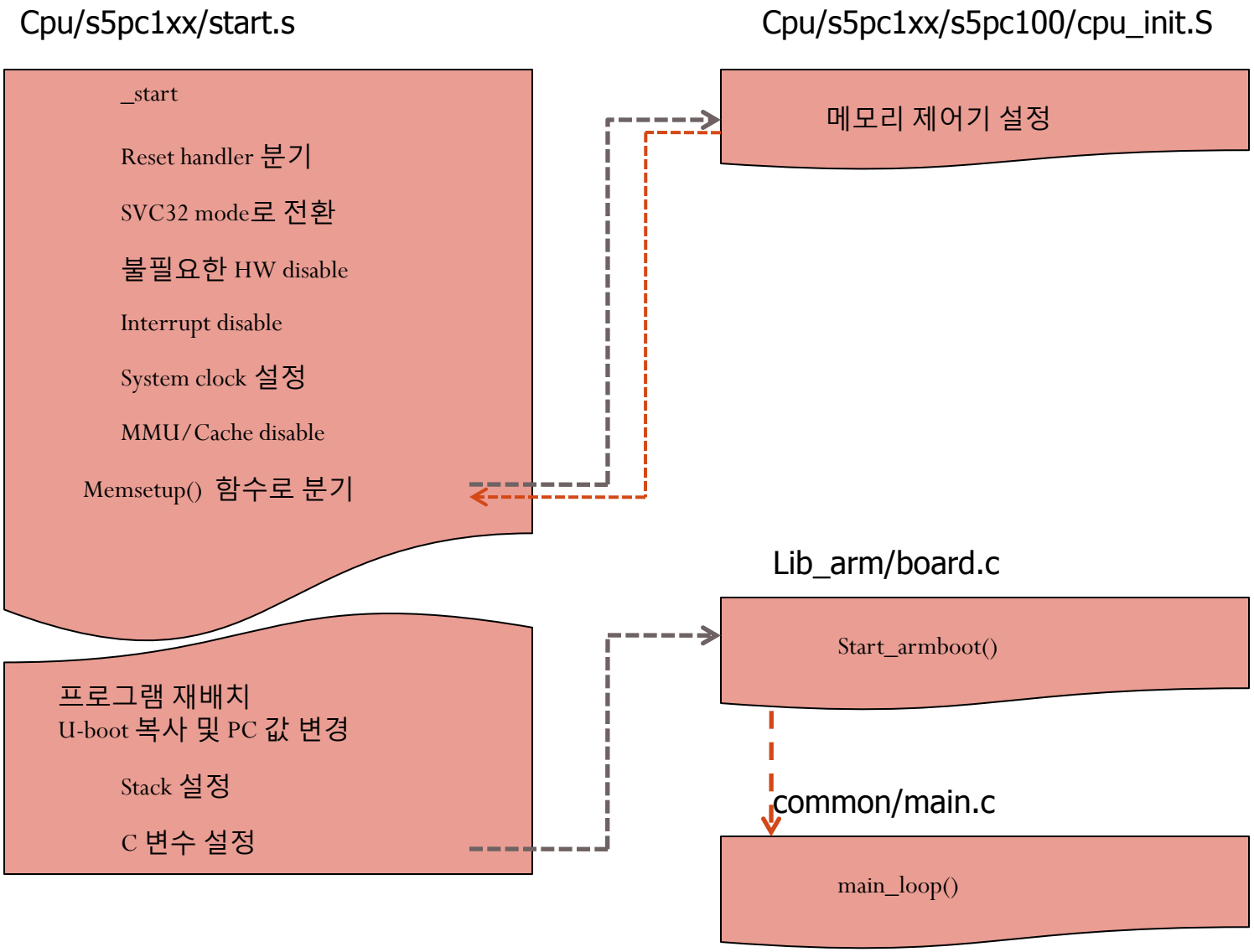
여기서 RAM의 _start_armboot 로 brach 하여 RAM에 있는 코드가 동작하게 된다.

Lib_arm/board.c 의 Start_armboot() 로 JUMP

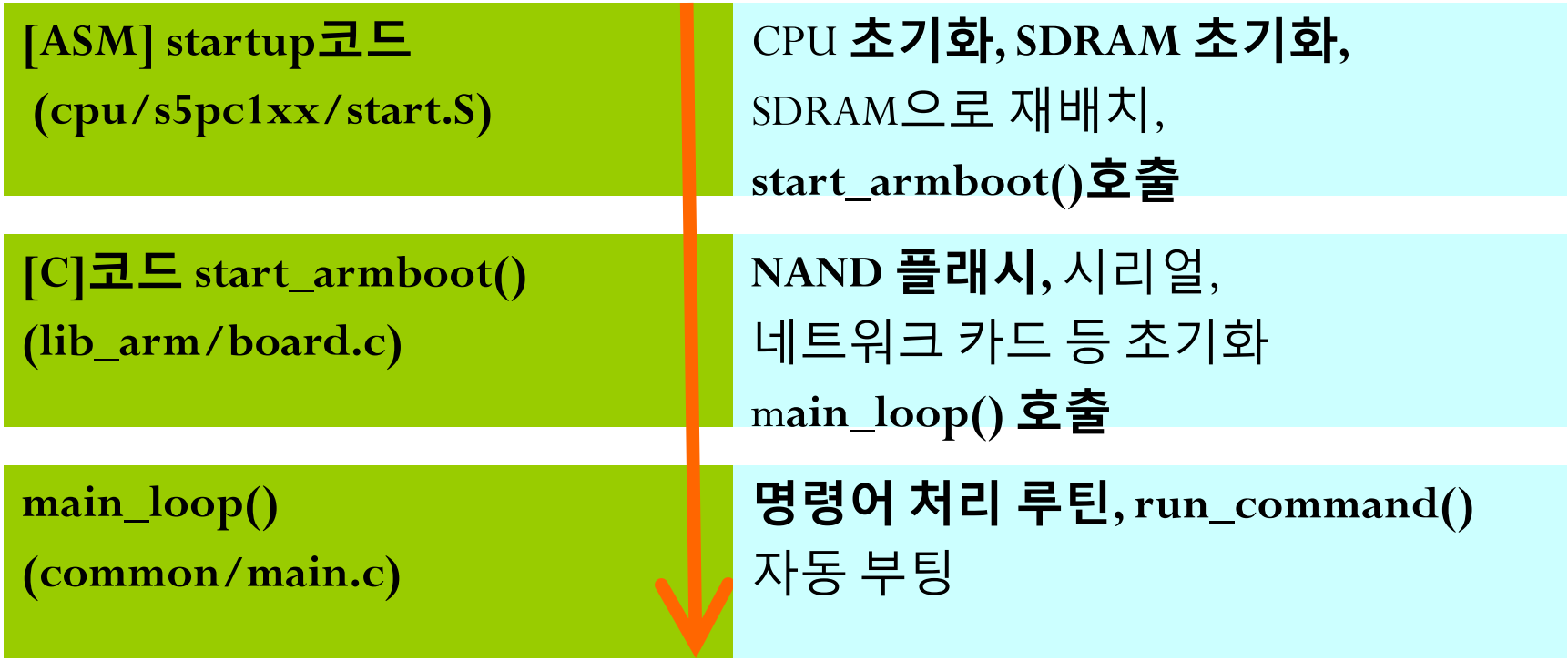
cpu/s5pc1xxx/start.s – setup the stack 후 memory map



U-boot 초기화 Diagram



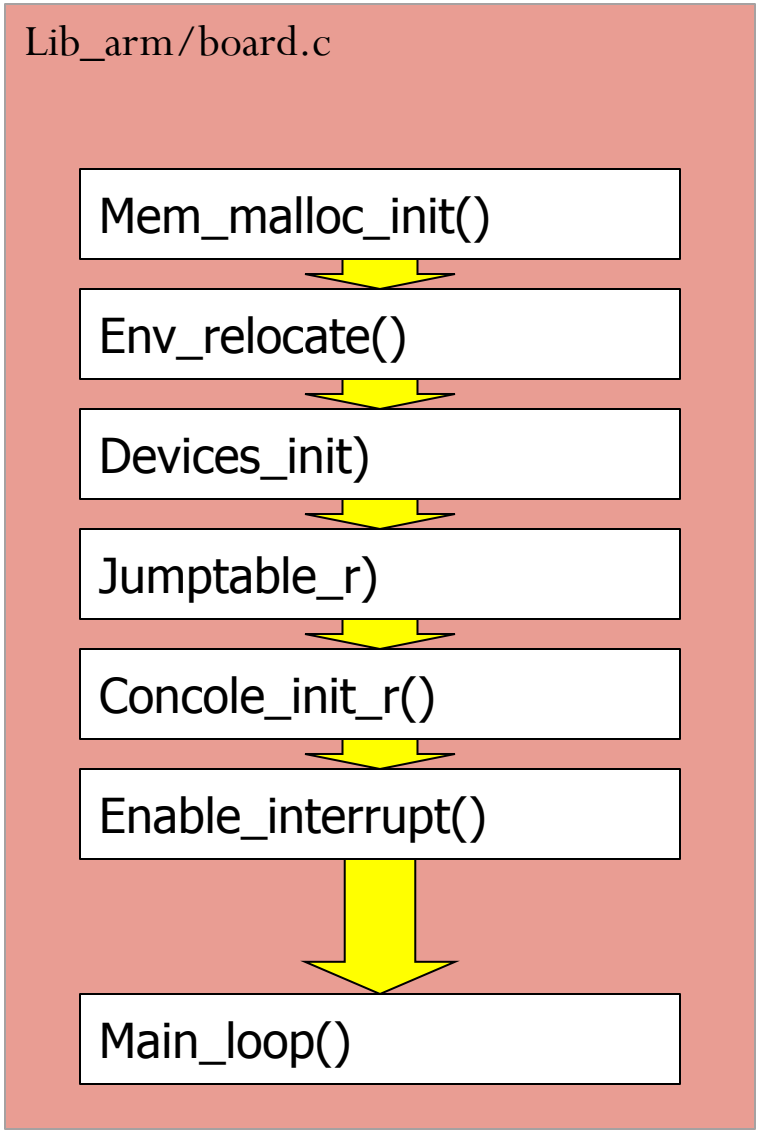
U-Boot 실행 순서 개요



Lib_arm/board.c – start_armboot()

```
init_fnc_t *init_sequence[] = {
        cpu_init,               /* basic cpu dependent setup */
#if defined(CONFIG_SKIP_RELOCATE_UBOOT)
        reloc_init,             /* Set the relocation done flag, must
                                   do this AFTER cpu_init(), but as soon
                                   as possible */
#endif
        board_init,             /* basic board dependent setup */
        interrupt_init,         /* set up exceptions */
        env_init,               /* initialize environment */
        init_baudrate,          /* initialze baudrate settings */
        serial_init,            /* serial communications setup */
        console_init_f,         /* stage 1 init of console */
        display_banner,         /* say that we are here */
#if defined(CONFIG_DISPLAY_CPUINFO)
        print_cpuinfo,          /* display cpu info (and speed) */
#endif
#if defined(CONFIG_DISPLAY_BOARDINFO)
        checkboard,             /* display board info */
#endif
#if defined(CONFIG_HARD_I2C) || defined(CONFIG_SOFT_I2C)
        init_func_i2c,
#endif
        dram_init,              /* configure available RAM banks */
        display_dram_config,
        NULL,
};
```

Lib_arm/board.c – start_armboot()



Lib_arm/board.c

Common/env_common.c

Common/devices.c

Common/exports.c

Common/console.c

Cpu/s5pc1xxx/interrupts.c

common/main.c

Main_loop()

```
    for (;;) {
#ifdef CONFIG_BOOT_RETRY_TIME
        if (rc >= 0) {
            /* Saw enough of a valid command to
             * restart the timeout.
             */
            reset_cmd_timeout();
        }
#endif
        len = readline (CFG_PROMPT);

        flag = 0;  /* assume no special flags for now */
        if (len > 0)
            strcpy (lastcommand, console_buffer);
        else if (len == 0)
            flag |= CMD_FLAG_REPEAT;
#ifdef CONFIG_BOOT_RETRY_TIME
        else if (len == -2) {
            /* -2 means timed out, retry autoboot
             */
            puts ("₩nTimed out waiting for command₩n");
# ifdef CONFIG_RESET_TO_RETRY
            /* Reinit board to run initialization code again */
            do_reset (NULL, 0, 0, NULL);
# else
            return;         /* retry autoboot */
# endif
        }
#endif

        if (len == -1)
            puts ("<INTERRUPT>₩n");
        else
            rc = run_command (lastcommand, flag);

        if (rc <= 0) {
            /* invalid command or not repeatable, forget it */
            lastcommand[0] = 0;
        }
    } ? end for ;; ?
```

Run_command()

```
/* find macros in this token and replace them */
process_macros (token, finaltoken);

/* Extract arguments */
argc = parse_line (finaltoken, argv);

/* Look up command in command table */
if ((cmdtp = find_cmd(argv[0])) == NULL) {
    printf ("Unknown command '%s' - try 'help'₩n", argv[0]);
    rc = -1;    /* give up after bad command */
    continue;
}

/* found - check max args */
if (argc > cmdtp->maxargs) {
    printf ("Usage:₩n%s₩n", cmdtp->usage);
    rc = -1;
    continue;
}

#if (CONFIG_COMMANDS & CFG_CMD_BOOTD)
    /* avoid "bootd" recursion */
    if (cmdtp->cmd == do_bootd) {
#ifdef DEBUG_PARSER
        printf ("[%s]₩n", finaltoken);
#endif
        if (flag & CMD_FLAG_BOOTD) {
            puts ("'bootd' recursion detected₩n");
            rc = -1;
            continue;
        }
        else
            flag |= CMD_FLAG_BOOTD;
    }
#endif  /* CFG_CMD_BOOTD */

    /* OK - call function to do the command */
    if ((cmdtp->cmd) (cmdtp, flag, argc, argv) != 0) {
        rc = -1;
    }
```

Command의 인자를 추출...

Command를 실행하기 위해 함수를 호출함

```
cmd_tbl_t *find_cmd (const char *cmd)
{
    cmd_tbl_t *cmdtp;
    cmd_tbl_t *cmdtp_temp = &__u_boot_cmd_start;    /*Init value */
    const char *p;
    int len;
    int n_found = 0;

    /*
     * Some commands allow length modifiers (like "cp.b");
     * compare command name only until first dot.
     */
    len = ((p = strchr(cmd, '.')) == NULL) ? strlen (cmd) : (p - cmd);

    for (cmdtp = &__u_boot_cmd_start;
        cmdtp != &__u_boot_cmd_end;
        cmdtp++) {
        if (strncmp (cmd, cmdtp->name,
            if (len == strlen (cmdtp->name))
                return cmdtp; /* full match */

            cmdtp_temp = cmdtp;   /* a
            n_found++;
        }
    }
    if (n_found == 1) {             /* exactly one match */
        return cmdtp_temp;
    }

    return NULL;  /* not found or ambiguous command */
} ? end find_cmd ?
```

Command list 에서 Command 와 일치하는 것을 Search

Match된 명령을 처리할 handle 함수 포인터를 return

U_BOOT_CMD

```
/*
 * Monitor Command Table
 */

struct cmd_tbl_s {
	char		*name;		/* Command Name			*/
	int		maxargs;	/* maximum number of arguments	*/
	int		repeatable;	/* autorepeat allowed?		*/
					/* Implementation function	*/
	int		(*cmd)(struct cmd_tbl_s *, int, int, char *[]);
	char		*usage;		/* Usage message	(short)	*/
#ifdef	CFG_LONGHELP
	char		*help;		/* Help  message	(long)	*/
#endif
#ifdef CONFIG_AUTO_COMPLETE
	/* do auto completion on the arguments */
	int		(*complete)(int argc, char *argv[], char last_char, int maxv, char *cmdv[]);
#endif
};

typedef struct cmd_tbl_s	cmd_tbl_t;
```

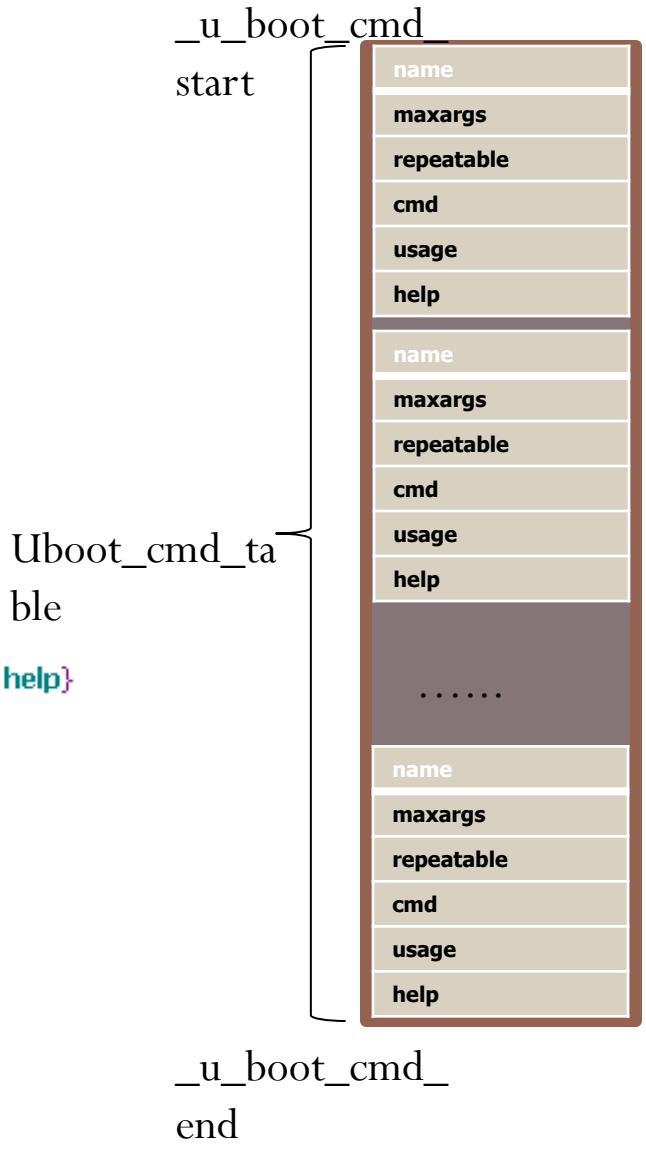
Example

```
int do_nfs (cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])
{
	return netboot_common(NFS, cmdtp, argc, argv);
}

U_BOOT_CMD(
	nfs,	3,	1,	do_nfs,
	"nfs₩t- boot image via network using NFS protocol₩n",
	"[loadAddress] [host ip addr:bootfilename]₩n"
);
```

U_BOOT_CMD

```
OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm",
"elf32-littlearm")
/*OUTPUT_FORMAT("elf32-arm", "elf32-arm", "elf32-
arm")*/
OUTPUT_ARCH(arm)
ENTRY(_start)
SECTIONS
 {
```

```
#define Struct_Section  __attribute__ ((unused,section (".u_boot_cmd")))

#ifdef  CFG_LONGHELP

#define U_BOOT_CMD(name,maxargs,rep,cmd,usage,help) ₩
cmd_tbl_t __u_boot_cmd_##name Struct_Section = {#name, maxargs, rep, cmd, usage, help}

#else       /* no long help info */

#define U_BOOT_CMD(name,maxargs,rep,cmd,usage,help) ₩
cmd_tbl_t __u_boot_cmd_##name Struct_Section = {#name, maxargs, rep, cmd, usage}

#endif    /* CFG_LONGHELP */
```



Boot_os_Fcn

```
/*
 *  Continue booting an OS image; caller already has:
 *  - copied image header to global variable `header'
 *  - checked header magic number, checksums (both header & image),
 *  - verified image architecture (PPC) and type (KERNEL or MULTI),
 *  - loaded (first part of) image to header load address,
 *  - disabled interrupts.
 */
typedef void boot_os_Fcn (cmd_tbl_t *cmdtp, int flag,
                          int  argc, char *argv[],
                          ulong     addr,           /* of image to boot */
                          ulong     *len_ptr,  /* multi-file image length table */
                          int  verify);     /* getenv("verify")[0] != 'n' */


#ifdef CONFIG_PPC
static boot_os_Fcn do_bootm_linux;
#else
extern boot_os_Fcn do_bootm_linux;
#endif
#ifdef CONFIG_SILENT_CONSOLE
static void fixup_silent_linux (void);
#endif
static boot_os_Fcn do_bootm_netbsd;
static boot_os_Fcn do_bootm_rtems;
#if (CONFIG_COMMANDS & CFG_CMD_ELF)
static boot_os_Fcn do_bootm_vxworks;
static boot_os_Fcn do_bootm_qnxelf;
int do_bootvx ( cmd_tbl_t *cmdtp, int flag, int argc, char *argv[] );
int do_bootelf (cmd_tbl_t *cmdtp, int flag, int argc, char *argv[] );
#endif /* CFG_CMD_ELF */
#if defined(CONFIG_ARTOS) && defined(CONFIG_PPC)
static boot_os_Fcn do_bootm_artos;
#endif
#ifdef CONFIG_LYNXKDI
static boot_os_Fcn do_bootm_lynxkdi;
extern void lynxkdi_boot( image_header_t * );
#endif
```

Lib_arm/armlinux.c

do_bootm() - 1

```
if (argc < 2) {
        addr = load_addr;
} else {
        addr = simple_strtoul(argv[1], NULL, 16);
}
```

Default Load address =0x20000000

Bootm 20008000 : loading 할 address 를 받아 command 로 받는다.

```
memmove (&header, (char *)addr, sizeof(image_header_t));
```

zImage의 헤더를 읽어온다.

```
typedef struct image_header {
        uint32_t  ih_magic;     /* Image Header Magic Number    */
        uint32_t  ih_hcrc;  /* Image Header CRC Checksum    */
        uint32_t  ih_time;  /* Image Creation Timestamp */
        uint32_t  ih_size;  /* Image Data Size          */
        uint32_t  ih_load;  /* Data     Load  Address       */
        uint32_t  ih_ep;          /* Entry Point Address          */
        uint32_t  ih_dcrc;  /* Image Data CRC Checksum */
        uint8_t         ih_os;          /* Operating System             */
        uint8_t         ih_arch;  /* CPU architecture         */
        uint8_t         ih_type; /* Image Type               */
        uint8_t         ih_comp;      /* Compression Type             */
        uint8_t         ih_name[IH_NMLEN];    /* Image Name           */
} image_header_t;
```

```
if (ntohl(hdr->ih_magic) != IH_MAGIC) {
        /* Check Magic Number */
}
```

```
data = (ulong)&header;
len  = sizeof(image_header_t);

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

if (crc32 (0, (char *)data, len) != checksum) {
        /* Check CRC32 */
}
```

```
data = addr + sizeof(image_header_t);
len  = ntohl(hdr->ih_size);
```

Pointer를 header에서 실제 압축된 커널을 가르킨다.

do_bootm() - 2

```
switch (hdr->ih_type) {
case IH_TYPE_STANDALONE:
      name = "Standalone Application";
      /* A second argument overwrites the load address */
      if (argc > 2) {
            hdr->ih_load = simple_strtoul(argv[2], NULL, 16);
      }
      break;
case IH_TYPE_KERNEL:
      name = "Kernel Image";
      break;
case IH_TYPE_MULTI:
      name = "Multi-File Image";
      len  = ntohl(len_ptr[0]);
      /* OS kernel is always the first image */
      data += 8; /* kernel_len + terminator */
      for (i=1; len_ptr[i]; ++i)
            data += 4;
      break;
default: printf ("Wrong Image Type for %s command₩n", cmdtp->name);
      SHOW_BOOT_PROGRESS (-5);
      return 1;
} ? end switch hdr->ih_type ?
```

LINUX의 경우는 IH_TYPE_KERNEL

Kernel + ramdisk 이미지

do_bootm() - 3

```
    switch (hdr->ih_comp) {
    case IH_COMP_NONE:
        if(ntohl(hdr->ih_load) == addr) {
            printf ("   XIP %s ... ", name);
        } else {
            memmove ((void *) ntohl(hdr->ih_load), (uchar *)data, len);
        }
        break;
    case IH_COMP_GZIP:
        printf ("   Uncompressing %s ... ", name);
        if (gunzip ((void *)ntohl(hdr->ih_load), unc_len,
                (uchar *)data, &len) != 0) {
            puts ("GUNZIP ERROR - must RESET board to recover₩n");
            SHOW_BOOT_PROGRESS (-6);
            do_reset (cmdtp, flag, argc, argv);
        }
        break;
#ifdef CONFIG_BZIP2
    case IH_COMP_BZIP2:
        printf ("   Uncompressing %s ... ", name);
        /*
         * If we've got less than 4 MB of malloc() space,
         * use slower decompression algorithm which requires
         * at most 2300 KB of memory.
         */
        i = BZ2_bzBuffToBuffDecompress ((char*)ntohl(hdr->ih_load),
                        &unc_len, (char *)data, len,
                        CFG_MALLOC_LEN < (4096 * 1024), 0);
        if (i != BZ_OK) {
            printf ("BUNZIP2 ERROR %d - must RESET board to recover₩n", i);
            SHOW_BOOT_PROGRESS (-6);
            udelay(100000);
            do_reset (cmdtp, flag, argc, argv);
        }
        break;
#endif /* CONFIG_BZIP2 */
    default:
        if (iflag)
            enable_interrupts();
        printf ("Unimplemented compression type %d₩n", hdr->ih_comp);
        SHOW_BOOT_PROGRESS (-7);
        return 1;
    } ? end switch hdr->ih_comp ?
```

압축된 커널을 압축 해제한다.

압축이 풀리는 위치는 (void *)ntohl(hdr_ih_load)이다

do_bootm() - 4

```
    switch (hdr->ih_os) {
    default:            /* handled by (original) Linux case */
    case IH_OS_LINUX:
#ifdef CONFIG_SILENT_CONSOLE
        fixup_silent_linux();
#endif
        do_bootm_linux  (cmdtp, flag, argc, argv,
                addr, len_ptr, verify);
        break;
    case IH_OS_NETBSD:
        do_bootm_netbsd (cmdtp, flag, argc, argv,
                addr, len_ptr, verify);
        break;

#ifdef CONFIG_LYNXKDI
    case IH_OS_LYNXOS:
        do_bootm_lynxkdi (cmdtp, flag, argc, argv,
                addr, len_ptr, verify);
        break;
#endif

    case IH_OS_RTEMS:
        do_bootm_rtems (cmdtp, flag, argc, argv,
                addr, len_ptr, verify);
        break;

#if (CONFIG_COMMANDS & CFG_CMD_ELF)
    case IH_OS_VXWORKS:
        do_bootm_vxworks (cmdtp, flag, argc, argv,
                addr, len_ptr, verify);
        break;
    case IH_OS_QNX:
        do_bootm_qnxelf (cmdtp, flag, argc, argv,
                addr, len_ptr, verify);
        break;
#endif /* CFG_CMD_ELF */
#ifdef CONFIG_ARTOS
    case IH_OS_ARTOS:
        do_bootm_artos  (cmdtp, flag, argc, argv,
                addr, len_ptr, verify);
        break;
#endif
    } ? end switch hdr->ih_os ?
```

드디어 Linux kernel 로 진입한다...!!!

do_bootm_linux()

```
void do_bootm_linux (cmd_tbl_t *cmdtp, int flag, int argc, char *argv[],
                     bootm_headers_t *images)
{
        ulong   initrd_start, initrd_end;
        ulong   ep = 0;
        bd_t    *bd = gd->bd;
        char    *s;
        int     machid = bd->bi_arch_number;
        void    (*theKernel)(int zero, int arch, uint params);
        int     ret;

#ifdef CONFIG_CMDLINE_TAG
        char *commandline = getenv ("bootargs");
#endif
```

Bootargs 를 가지고 온다

```
 theKernel = (void (*)(int, int, uint))ep;

 s = getenv ("machid");
 if (s) {
         machid = simple_strtoul (s, NULL, 16);
         printf ("Using machid 0x%x from environment\n", machid);
 }

 ret = boot_get_ramdisk (argc, argv, images, IH_ARCH_ARM,
                 &initrd_start, &initrd_end);
 if (ret)
         goto error;
```

Kernel에게 이미지 entry poiboot_get_ramdisknt를 저

Machin id를 가지고 온다

Ram disk가 있는지 없는지 확인

do_bootm_linux()

```
cleanup_before_linux ();

theKernel (0, machid, bd->bi_boot_params);
/* does not return */
return;
```

Kernel image로 제어권을 넘겨준다.

```
int cleanup_before_linux (void)
{
        /*
         * this function is called just before we call linux
         * it prepares the processor for linux
         *
         * we turn off caches etc ...
         */

        unsigned long i;

        disable_interrupts ();

        /* turn off I/D-cache */
        asm ("mrc p15, 0, %0, c1, c0, 0":"=r" (i));
        i &= ~(C1_DC | C1_IC);
        asm ("mcr p15, 0, %0, c1, c0, 0": :"r" (i));

        /* invalidate I-cache */
        arm_cache_flush();

        i = 0;
        /* mem barrier to sync up things */
        asm("mcr p15, 0, %0, c7, c10, 4": :"r"(i));

        return(0);
}
```