

# Head.S 분석자료

16기 A조

Head.s @0 stext

X0	DTB (PHY ADDRERSS)		
X1	00000000_00000000	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	FP
X15		X30	LR

```
bl    preserve_boot_args
```

X0	DTB (PHY ADDRERSS)		
X1	00000000_00000000	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+8

FP

LR

```
mov    x21, x0
```

X0	DTB (PHY ADDRERSS)		
X1	00000000_00000000	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+8

FP

LR

Head.s @3 preserve\_boot\_args

adr\_l x0, boot\_args

u64 \_\_cacheline\_aligned boot\_args[4];

X0	&boot_args		
X1	00000000_00000000	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+8

FP

LR

Head.s @4 preserve\_boot\_args

```
stp    x21, x1, [x0]
stp    x2, x3, [x0, #16]

u64 __cacheline_aligned boot_args[4];

boot_args[0] = x21;    // DTB
boot_args[1] = x1;     // 0
boot_args[2] = x2;     // 0
boot_args[3] = x3;     // 0
```

X0	&boot_args		
X1	00000000_00000000	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+8

FP

LR

dmb sy

**DMB**  
Data Memory Barrier.

**SY**  
Full system barrier operation. This is the default and can be omitted.

boot\_arg[]를 저장한 명령어가  
Chche clear 명령어에 영향을  
받지 않도록 Barrier를 사용한다.  
(메모리 오더링 이슈는 1 core에서도 발생한다)

X0	&boot_args		
X1	00000000_00000000	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+8

FP

LR

Head.s @6 preserve\_boot\_args

```
mov    x1, #0x20
```

4 x 8 = 32 = 0x20

X0	&boot_args		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+8

FP

LR



b     \_\_inval\_dcache\_area

\_\_inval\_dcache\_area(void \*addr, size\_t len);

X0 – addr

X1 – len

Ensure that any D-cache lines are  
invalidated.

X0	&boot_args		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+8

FP

LR

Head.s @8 stext

bl el2\_setup

X0	&boot_args		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP

LR

msr SPsel, #1

X0	&boot_args		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP

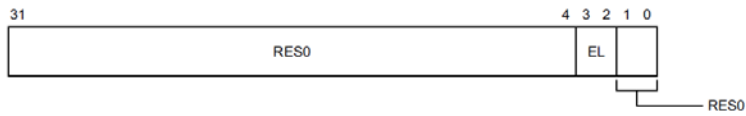
LR

SPSEL
00000000_000000001

mrs x0, CurrentEL

System이 EL1 모드로  
부팅했다고 가정함

The CurrentEL bit assignments are:



**Bits [31:4]**  
Reserved, RES0.

**EL, bits [3:2]**  
Current exception level. Possible values of this field are:

00	EL0
01	EL1
10	EL2
11	EL3

Resets to an IMPLEMENTATION DEFINED value.

**Bits [1:0]**  
Reserved, RES0.

X0	00000000_000000004		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP  
LR

SPSEL
00000000_00000001

cmp x0, #CurrentEL\_EL2 (=8)

```
/* Current Exception Level values, as contained in CurrentEL */
#define CurrentEL_EL1  (1 << 2)
#define CurrentEL_EL2  (2 << 2)
```

X0	00000000_00000004		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

SPSEL

00000000\_00000001

FP

LR

b.eq 1f

같지 않으므로 skip

X0	00000000_00000004		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP

LR

SPSEL
00000000_00000001

```
movq x0, (SCTLR_EL1_RES1 | ENDIAN_SET_EL1)
(SCTLR_EL1_RES1 | ENDIAN_SET_EL1) -> 0x30500800
```

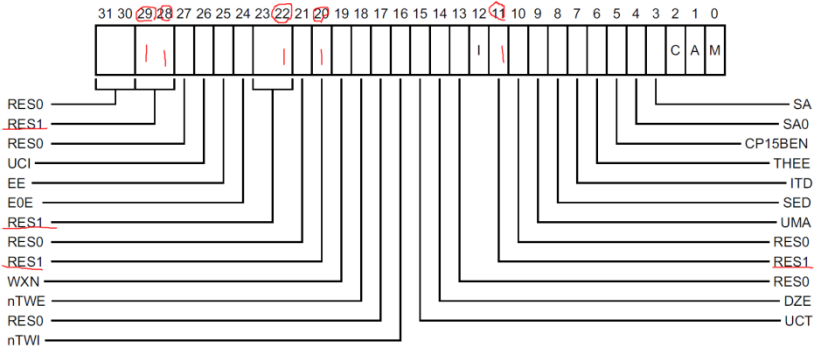
X0	00000000_30500800		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP  
LR

SPSEL
00000000_00000001

```
msr    sctlr_el1, x0
```

Reserved 된 값을 1로 채움  
왜 하는지는 정확히 모름



X0	00000000_30500800		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP  
LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800



```
mov    w0, #BOOT_CPU_MODE_EL1 (=0xe11)
```

```
#define BOOT_CPU_MODE_EL1→  (0xe11)
#define BOOT_CPU_MODE_EL2→  (0xe12)
```

X0	00000000_00000E11		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

ret

stext로 return 함

X0	00000000_00000E11		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

adrp x23, \_\_PHYS\_OFFSET

KERNEL\_START = ffff000010080000

TEXT\_OFFSET = 0x80000

\_\_PHYS\_OFFSET = ffff000010000000

#define \_\_PHYS\_OFFSET (KERNEL\_START - TEXT\_OFFSET)

X0	00000000_00000E11		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	FFFF0000_10000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

and x23, x23, MIN\_KIMG\_ALIGN - 1

```
/*
 * arm64 requires the kernel image to placed
 * TEXT_OFFSET bytes beyond a 2 MB aligned base
 */
#define MIN_KIMG_ALIGN SZ_2M
```

```
#define SZ_1M 0x00100000
#define SZ_2M 0x00200000
#define SZ_4M 0x00400000
#define SZ_8M 0x00800000
#define SZ_16M 0x01000000
#define SZ_32M 0x02000000
```

X0	00000000_00000E11		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+16

FP  
LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

bl set\_cpu\_boot\_mode\_flag

X0	00000000_00000E11		
X1	00000000_00000020	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+40

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

Head.s @20 set\_cpu\_boot\_mode\_flag

adr\_l x1, \_\_boot\_cpu\_mode

X0	00000000_00000E11		
X1	& __boot_cpu_mode	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+40

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

Head.s @21 set\_cpu\_boot\_mode\_flag

cmp w0, #BOOT\_CPU\_MODE\_EL2

```
#define BOOT_CPU_MODE_EL1 (0xe11)
#define BOOT_CPU_MODE_EL2 (0xe12)
```

X0	00000000_00000E11		
X1	& __boot_cpu_mode	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+40

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

Head.s @22 set\_cpu\_boot\_mode\_flag

b.ne 1f

0xE11 != 0xE12 -> b 1f

X0	00000000_00000E11		
X1	& __boot_cpu_mode	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+40

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800



str w0, [x1]

\*x1 = w0;  
\_\_boot\_cpu\_mode = 0x00000E11

X0	00000000_00000E11		
X1	& __boot_cpu_mode	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+40

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

```
dmb sy
dc ivac, x1
```

\_\_boot\_cpu\_mode에 값을 저장하고 코어내의 boot\_cpu\_mode 데이터 캐시 라인(PoC 관점) 을 무효화 시킨다.

DC IVAC Invalidate by Virtual Address, to Point of Coherency

X0	00000000_00000E11		
X1	& __boot_cpu_mode	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+40

FP

LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

bl    \_\_create\_page\_tables

X0	00000000_00000E11		
X1	& __boot_cpu_mode	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	
X14		X29	
X15		X30	stext+48

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

mov x28, lr

X0	00000000_00000E11		
X1	& __boot_cpu_mode	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	stext+48

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

```
adrp x0, init_pg_dir
adrp x1, init_pg_end
```

X0	& init_pg_dir		
X1	& init_pg_end	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	stext+48

FP  
LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

```
sub    x1, x1, x0
```

init\_pg\_dir ~ init\_pg\_end 까지 크기

```
extern pgd_t init_pg_dir[PTRS_PER_PGD];
extern pgd_t init_pg_end[];
extern pgd_t swapper_pg_dir[PTRS_PER_PGD];
extern pgd_t idmap_pg_dir[PTRS_PER_PGD];
extern pgd_t tramp_pg_dir[PTRS_PER_PGD];
```

```
    . = ALIGN(PAGE_SIZE);
    init_pg_dir = .;
    . += INIT_DIR_SIZE;
    init_pg_end = .;
```

X0	& init_pg_dir		
X1	INIT_DIR_SIZE	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	stext+48

FP  
LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

bl \_\_inval\_dcache\_area

Init\_pg\_dir 부터 INIT\_DIR\_SIZE 만큼  
데이터 캐시를 무효화 시킨다.

X0	& init_pg_dir		
X1	INIT_DIR_SIZE	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	@29 + 8

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

```
adrp x0, init_pg_dir
adrp x1, init_pg_end
```

X0	& init_pg_dir		
X1	& init_pg_end	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	@29 + 8

FP  
LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800



```
sub    x1, x1, x0
```

init\_pg\_dir ~ init\_pg\_end 까지 크기

```
extern pgd_t init_pg_dir[PTRS_PER_PGD];
extern pgd_t init_pg_end[];
extern pgd_t swapper_pg_dir[PTRS_PER_PGD];
extern pgd_t idmap_pg_dir[PTRS_PER_PGD];
extern pgd_t tramp_pg_dir[PTRS_PER_PGD];
```

```
    . = ALIGN(PAGE_SIZE);
    init_pg_dir = .;
    . += INIT_DIR_SIZE;
    init_pg_end = .;
```

X0	& init_pg_dir		
X1	INIT_DIR_SIZE	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	@29 + 8

FP  
LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

```
stp xzr, xzr, [x0], #16
stp xzr, xzr, [x0], #16
stp xzr, xzr, [x0], #16
stp xzr, xzr, [x0], #16
```

16 \* 4 = 64 bytes  
memset(init\_pg\_dir, 0, 64)  
Init\_pg\_dir 부터 64byte 씩 0으로 초기화 한다.

```
stp      xzr, xzr, [x0, #16]      preindex

        x0 += 16
        *(x0  ) = xzr
        *(x0 + 8) = xzr

stp      xzr, xzr, [x0], #16      postindex

        *(x0  ) = xzr
        *(x0 + 8) = xzr
        x0 += 16
```

X0	& init_pg_dir + 64		
X1	INIT_DIR_SIZE	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	@29 + 8

FP  
LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

```
subs      x1, x1, #64
b.ne      1b
```

X1 값이 0이 아니면 레이블 1(@32)로 분기  
X1 값이 0이면 다음 라인(@34) 실행

X0	& init_pg_dir + 64		
X1	INIT_DIR_SIZE - 64	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7		X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	@29 + 8

FP

LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800

```
mov    x7, SWAPPER_MM_MMUFLAGS
```

SWAPPER\_MM\_MMUFLAGS  
= (((4) << 2) | ((3 << 0) | (1 << 10) | (3 << 8)))  
= 0x713

X0	& init_pg_dir + 64		
X1	INIT_DIR_SIZE - 64	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7	00000000_00000713	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+48
X14		X29	
X15		X30	@29 + 8

FP  
LR

SPSEL
00000000_00000001

SCTLR_EL1
00000000_30500800