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

Head.s @1 stext

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	FP
X15		X30	stext+8 LR

Head.s @2 preserve_boot_args

```
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	FP
X15		X30	stext+8 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	FP
X15		X30	stext+8 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		FP
X15		X30	stext+8	LR

dmb sy

DMB
Data Memory Barrier.

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

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		FP
X15		X30	stext+8	LR

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

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		FP
X15		X30	stext+8	LR

Head.s @7 preserve_boot_args

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		FP
X15		X30	stext+8	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		FP
X15		X30	stext+16	LR

Head.s @9 el2_setup

```
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	FP
X15		X30	stext+16 LR

SPSEL
00000000_00000001

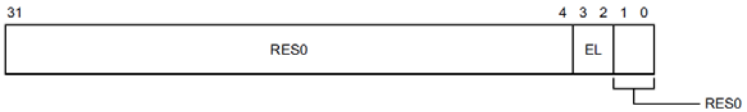
mrs x0, CurrentEL

System이 EL1 모드로 부팅했다고 가정함

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	FP
X15		X30	stext+16 LR

SPSEL
00000000_00000001

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.

Head.s @11 el2_setup

cmp x0, #CurrentEL_EL2 (=8)

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	FP
X15		X30	stext+16 LR

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

SPSEL
00000000_00000001

Head.s @12 el2_setup

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	FP
X15		X30	stext+16 LR

SPSEL
00000000_00000001

Head.s @13 el2_setup

mov_q x0, (SCTLR_EL1_RES1 | ENDIAN_SET_EL1)

(SCTLR_EL1_RES1 | ENDIAN_SET_EL1) -> 0x30500800

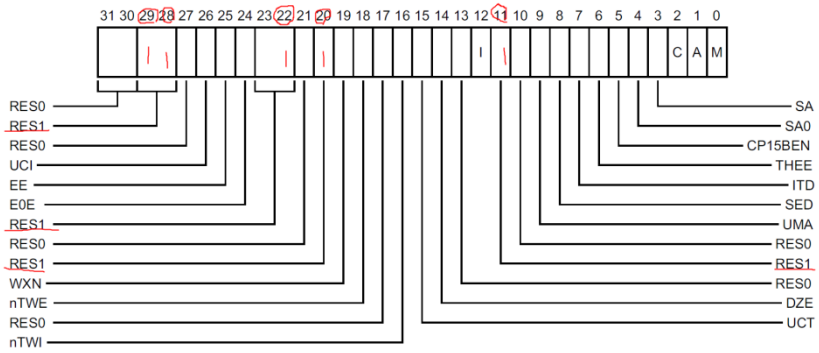
X0	00000000_30500800			
X1	00000000_00000020	X16		
X2	00000000_00000000	X17		
X3	00000000_00000000	X18		
X4		X19		
X5		X20		
X6		X21	DTB (PHY ADDRERSS)	
X7		X22		
X8		X23		
X9		X24		
X10		X25		
X11		X26		
X12		X27		
X13		X28		
X14		X29		FP
X15		X30	stext+16	LR

SPSEL
00000000_00000001

Head.s @14 el2_setup

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	FP
X15		X30	stext+16 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @15 el2_setup

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	FP
X15		X30	stext+16 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @16 el2_setup

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		FP
X15		X30	stext+16	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @17 stext

```
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		FP
X15		X30	stext+16	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @18 stext

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	FP
X15		X30	stext+16 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @19 stext

```
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	FP
X15		X30	stext+40 LR

SPSEL	SCTLR_EL1
00000000_00000001	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	FP
X15		X30	stext+40 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @21 set_cpu_boot_mode_flag

```
cmp     w0, #BOOT_CPU_MODE_EL2
```

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	FP
X15		X30	stext+40 LR

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

SPSEL	SCTLR_EL1
00000000_00000001	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		FP
X15		X30	stext+40	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @23 set_cpu_boot_mode_flag

str w0, [x1]

*x1 = w0;
__boot_cpu_mode = 0x**00000E11**

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		FP
X15		X30	stext+40	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @24 set_cpu_boot_mode_flag

```
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		FP
X15		X30	stext+40	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @25 stext

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	FP
X15		X30	stext+24 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @26 __create_page_tables

```
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+24
X14		X29	FP
X15		X30	stext+24 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @27 __create_page_tables

```
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+24
X14		X29	FP
X15		X30	stext+24 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @28 __create_page_tables

```
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+24	
X14		X29		FP
X15		X30	stext+24	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @29 __create_page_tables

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+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @30 __create_page_tables

```
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+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @31 __create_page_tables

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+24	
X14		X29		FP
X15		X30	@29 + 8	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @32 __create_page_tables

```
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+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @33 __create_page_tables

```
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+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @34 __create_page_tables

mov x7, SWAPPER_MM_MMUFLAGS

#define SWAPPER_MM_MMUFLAGS
(PMD_ATTRINDX(MT_NORMAL) | SWAPPER_PMD_FLAGS)

#define MT_NORMAL 4
#define PMD_ATTRINDX(*t*) (_AT(**pmdval_t**, (t)) << 2)

#define SWAPPER_PMD_FLAGS
(PMD_TYPE_SECT | PMD_SECT_AF | PMD_SECT_S)

PMD_ATTRINDX(MT_NORMAL) : (4 << 2)
PMD_TYPE_SECT : (1 << 0)
PMD_SECT_AF : (1 << 10)
PMD_SECT_S : (3 << 8)

SWAPPER_MM_MMUFLAGS
= (4 << 2) | ((1 << 0) | (1 << 10) | (3 << 8)))
= 0x711

X0	& init_pg_dir + 64		
X1	0	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @35 __create_page_tables

adrp x0, idmap_pg_dir

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	00000000_00000000	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @36 __create_page_tables

adrp x3, __idmap_text_start

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	&__idmap_test_start	X18	
X4		X19	
X5		X20	
X6		X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @37 __create_page_tables

```
mov     x5, #VA_BITS
```

VA_BITS 48

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	&__idmap_test_start	X18	
X4		X19	
X5	00000000_00000030	X20	
X6		X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @37 __create_page_tables

```
mov     x5, #VA_BITS
```

VA_BITS 48

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	&__idmap_test_start	X18	
X4		X19	
X5	00000000_00000030	X20	
X6		X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @38 __create_page_tables

```
adr_l       x6, vabits_user
str         x5, [x6]
dmb         sy
dc          ivac, x6
```

```
u64 vabits_user;
```

```
adr_l       x6, vabits_user
str         x5, [x6]
```
vabits_user = 0x30;

```
dmb         sy
dc          ivac, x6
```
vabits_user 에 값을 저장하고 코어내의 vabits_user 데이터 캐시 라인(PoC 관점) 을 무효화 시킨다.

X0	& idmap_pg_dir			
X1	0	X16		
X2	00000000_00000000	X17		
X3	&__idmap_test_start	X18		
X4		X19		
X5	00000000_00000030	X20		
X6	&vabits_user	X21	DTB (PHY ADDRERSS)	
X7	00000000_00000711	X22		
X8		X23	00000000_00000000	
X9		X24		
X10		X25		
X11		X26		
X12		X27		
X13		X28	stext+24	
X14		X29		FP
X15		X30	@29 + 8	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @39 __create_page_tables

```
adrp        x5, __idmap_text_end
clz         x5, x5
cmp         x5, TCR_T0SZ(VA_BITS) <= 16
b.ge        1f
```

1. X5 __idmap_text_end symbol 로딩
2. 값은 ffff000010b8d658
3. CLZ X5 X5 -> 0으로 최종 저장 됨

__idmap_text_end ffff000010b8d658

TCR_T0SZ(VA_BITS) ((UL(64) – 48) << 0) = 16

b.ge 1f (f : forward, b : backward)

@40 으로 감.

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	&__idmap_test_start	X18	
X4		X19	
X5	0	X20	
X6	&vabits_user	X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

```
#define TCR_T0SZ_OFFSET        0
#define TCR_T1SZ_OFFSET        16
#define TCR_T0SZ(x)            ((UL(64) - (x)) << TCR_T0SZ_OFFSET)
#define TCR_T1SZ(x)            ((UL(64) - (x)) << TCR_T1SZ_OFFSET)
#define TCR_TxSZ(x)            (TCR_T0SZ(x) | TCR_T1SZ(x))
```

Head.s @40 __create_page_tables

```
adr_l       x6, idmap_t0sz
str         x5, [x6]
dmb         sy
dc          ivac, x6
```

*(&idmap_t0sz) = 0;

idmap_t0sz 에 값을 저장하고 코어내의 idmap_t0sz 데이터 캐시 라인(PoC 관점) 을 무효화 시킨다.

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	&__idmap_test_start	X18	
X4		X19	
X5	0	X20	
X6	&idmap_t0sz	X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @41 __create_page_tables

```
mov             x4, #1 << (PHYS_MASK_SHIFT - PGDIR_SHIFT)
str_l           x4, idmap_ptrs_per_pgd, x5
```

```
#define PHYS_MASK_SHIFT          (48)
#define CONFIG_PGTABLE_LEVELS 4
#define PGDIR_SHIFT  ARM64_HW_PGTABLE_LEVEL_SHIFT(4 - CONFIG_PGTABLE_LEVELS)

#define ARM64_HW_PGTABLE_LEVEL_SHIFT(n) ((PAGE_SHIFT - 3) * (4 - (n)) + 3)

#define PAGE_SHIFT 12
#define ARM64_HW_PGTABLE_LEVEL_SHIFT(n) ((PAGE_SHIFT - 3) * (4 - (n)) + 3)
(PHYS_MASK_SHIFT - PGDIR_SHIFT) = 39
```

*(&idmap_ptrs_per_pgd) = **00000080_00000000**

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	&__idmap_test_start	X18	
X4	00000080_00000000	X19	
X5	& idmap_ptrs_per_pgd	X20	
X6	&idmap_t0sz	X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @42 __create_page_tables

```
ldr_l       x4, idmap_ptrs_per_pgd
mov         x5, x3
adr_l       x6, __idmap_text_end
```

X0	& idmap_pg_dir		
X1	0	X16	
X2	00000000_00000000	X17	
X3	&__idmap_test_start	X18	
X4	& idmap_ptrs_per_pgd	X19	
X5	&__idmap_test_start	X20	
X6	& __idmap_text_end	X21	DTB (PHY ADDRERSS)
X7	00000000_00000711	X22	
X8		X23	00000000_00000000
X9		X24	
X10		X25	
X11		X26	
X12		X27	
X13		X28	stext+24
X14		X29	FP
X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @43 __create_page_tables

map_memory x0, x1, x3, x6, x7, x3, x4, x10, x11, x12, x13, x14

* Map memory for specified virtual address range. Each level of page table needed supports
* multiple entries. If a level requires n entries the next page table level is assumed to be
* formed from n pages.
*
* tbl: location of page table
* rtbl: address to be used for first level page table entry (typically tbl + PAGE_SIZE)
* vstart: start address to map
* vend: end address to map - we map [vstart, vend]
* flags: flags to use to map last level entries
* phys: physical address corresponding to vstart - physical memory is contiguous
* pgds: the number of pgd entries
*
* Temporaries: istart, iend, tmp, count, sv - these need to be different registers
* Preserves: vstart, vend, flags
* Corrupts: tbl, rtbl, istart, iend, tmp, count, sv

.macro map_memory, tbl, rtbl, vstart, vend, flags, phys, pgds, istart, iend, tmp, count, sv

tbl	X0	& idmap_pg_dir			
rtbl	X1	0	X16		
	X2	00000000_00000000	X17		
phys vstart	X3	&__idmap_text_start	X18		
pgds	X4	& idmap_ptrs_per_pgd	X19		
	X5	&__idmap_text_start	X20		
vend	X6	& __idmap_text_end	X21	DTB (PHY ADDRERSS)	
flags	X7	00000000_00000711	X22		
	X8		X23	00000000_00000000	
	X9		X24		
istart	X10		X25		
iend	X11		X26		
tmp	X12		X27		
count	X13		X28	stext+24	
sv	X14		X29		FP
	X15		X30	@29 + 8	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @44 __create_page_tables

```
adrp        x0, init_pg_dir
mov_q       x5, KIMAGE_VADDR + TEXT_OFFSET
            // compile time __va(_text)
add         x5, x5, x23
            // add KASLR displacement
mov         x4, PTRS_PER_PGD
adrp        x6, _end
            // runtime __pa(_end)
adrp        x3, _text
            // runtime __pa(_text)
sub         x6, x6, x3
            // _end - _text
add         x6, x6, x5
            // runtime __va(_end)

map_memory x0, x1, x5, x6, x7, x3, x4, x10, x11, x12, x13, x14
```

#define PTRS_PER_PGD (1 << (48- 39(@41))) = 512

tbl	X0	& init_pg_dir		
rtbl	X1	0	X16	
	X2	00000000_00000000	X17	
phys	X3	&_text	X18	
pgds	X4	00000000_00000200	X19	
vstart	X5	&_text	X20	
vend	X6	&_end	X21	DTB (PHY ADDRERSS)
flags	X7	00000000_00000711	X22	
	X8		X23	00000000_00000000
	X9		X24	
istart	X10		X25	
iend	X11		X26	
tmp	X12		X27	
count	X13		X28	stext+24
sv	X14		X29	FP
	X15		X30	@29 + 8 LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800

Head.s @45 __create_page_tables

```
adrp    x0, idmap_pg_dir
adrp    x1, init_pg_end
sub     x1, x1, x0
dmb     sy
bl      __inval_dcache_area

ret     x28
```

__inval_dcache_area(void *addr, size_t len);

X0 – addr
X1 – len

Ensure that any D-cache lines are invalidated.

stext+24 주소로 return

tbl	X0	& idmap_pg_dir			
rtbl	X1	Sizeof(idmap_pg_dir +init_pg_dir)	X16		
	X2	00000000_00000000	X17		
phys	X3	&_text	X18		
pgds	X4	00000000_00000200	X19		
vstart	X5	&_text	X20		
vend	X6	&_end	X21	DTB (PHY ADDRERSS)	
flags	X7	00000000_00000711	X22		
	X8		X23	00000000_00000000	
	X9		X24		
istart	X10		X25		
iend	X11		X26		
tmp	X12		X27		
count	X13		X28	stext+24	
sv	X14		X29		FP
	X15		X30	&(ret x28)	LR

SPSEL	SCTLR_EL1
00000000_00000001	00000000_30500800