pegmatite_smp_prepare_cpus()

```
void __init smp_prepare_cpus(unsigned int max_cpus)
{
        unsigned int ncores = num_possible_cpus();

        init_cpu_topology();

        smp_store_cpu_info(smp_processor_id());

        /*
         * are we trying to boot more cores than exist?
         */
        if (max_cpus > ncores)
                max_cpus = ncores;
        if (ncores > 1 && max_cpus) {
                /*
                 * Initialise the present map, which describes the set of CPUs
                 * actually populated at the present time. A platform should
                 * re-initialize the map in the platforms smp_prepare_cpus()
                 * if present != possible (e.g. physical hotplug).
                 */
                init_cpu_present(cpu_possible_mask);

                /*
                 * Initialise the SCU if there are more than one CPU
                 * and let them know where to start.
                 */
                if (smp_ops.smp_prepare_cpus)
                        smp_ops.smp_prepare_cpus(max_cpus);
        }
}
```

核心是smp_ops.smp_prepare_cpus(max_cpus)，也就是pegmatite_smp_prepare_cpus(max_cpus)

```c
static void __init pegmatite_smp_prepare_cpus(unsigned int max_cpus)
{
        struct device_node *node;
        struct resource res;
        int ret;
        int cpu;
        void __iomem *squ_addr = NULL;

        node = of_find_compatible_node(NULL, NULL, "marvell,pegmatite-smpboot-sram");    ①
        if (!node) {
                pr_err("%s: could not find sram dt node\n", __func__);
                goto err;
        }
        ret = of_address_to_resource(node, 0, &res);        ②
        if (ret < 0) {
                pr_err("%s: could not get address for node %s\n",
                       __func__, node->full_name);
                goto err;
        }

        if (resource_size(&res) < pegmatite_smp_jump_size) {        ③
                pr_err("%s: invalid sram reservation\n", __func__);
                goto err;
        }

        /*
         * pegmatite_smp_jump includes the instructions needed to get us from
         * the A53's reset vector to pegmatite_secondary_startup.
         *
         * The jump address is the 4 bytes immediately after it, referenced by
         * pegmatite_smp_jump_address.
         *
         * pegmatite_smp_jump and the address of pegmatite_secondary_startup
         * are copied to the cpu's reset vector at 0xd1000000.  This address
         * is the first bank of the SQU.
         *
         * Write the address of pegmatite_secondary_startup before copying the
         * section containing the code and the address.
         *
         */

        squ_addr = of_iomap(node, 0);        ④
        if (!squ_addr)
                goto err;

        pegmatite_boot_addr = kzalloc(        ⑤
                sizeof(pegmatite_boot_addr[0]) * num_present_cpus(),
                GFP_KERNEL);
        if (!pegmatite_boot_addr) {
                pr_err("Failed to allocate CPU jump table\n");
                goto err;
        }

        pegmatite_smp_jump_table = virt_to_phys(pegmatite_boot_addr);   ⑥

        /* Copy the jump instructions from pegmatite_smp_jump to the SQU */
        memcpy(squ_addr, &pegmatite_smp_jump, pegmatite_smp_jump_size); ⑦
        wmb();
                                                ⑧
        iounmap(squ_addr);


        return;
err:
        if (squ_addr)
                iounmap(squ_addr);

        for_each_present_cpu(cpu) {
                if (cpu == smp_processor_id())
                        continue;
                set_cpu_present(cpu, 0);
                pr_warn("%s: Disabling SMP\n", __func__);
        }
}
```

①

从device tree中获得如下device_node

```
                smpboot-sram@0 {
                        compatible = "marvell,pegmatite-smpboot-sram";
                        reg = <0x0 0x20>;
                };
```

②

获得reg = <0x0 0x20> property

res.start = 0xd1000000

res.end = 0xd100001f

③

确定指定的buffer size > pegmatite_smp_jump_size

这里的pegmatite_smp_jump_size见drivers/platform/pegmatite/smp/headsmp.S

```
ENTRY(pegmatite_smp_jump)
        ldr     r1, pegmatite_smp_jump_table
        mrc     p15, 0, r0, c0, c0, 5   @ Read MPIDR
        and     r0, r0, #0xf            @ Get CPU number in R0
        ldr     r1, [r1, r0, LSL #2]    @ Read jump address for this cpu
        blx     r1

        .globl  pegmatite_smp_jump_table
pegmatite_smp_jump_table:
        .long   0x0
ENDPROC(pegmatite_smp_jump)
        .globl pegmatite_smp_jump_size
pegmatite_smp_jump_size:
        .long   . - pegmatite_smp_jump
```

这一小段code是secondary core起来时要执行的code.

从source code上看不出上面的code占多少bytes，通过disassemble vmlinux就可以很容易知道大小了。

```
c04423f8 <pegmatite_smp_jump>:
c04423f8:       e59f100c        ldr     r1, [pc, #12]   ; c044240c <pegmatite_smp_jump_table>
c04423fc:       ee100fb0        mrc     15, 0, r0, cr0, cr0, {5}
c0442400:       e200000f        and     r0, r0, #15
c0442404:       e7911100        ldr     r1, [r1, r0, lsl #2]
c0442408:       e12fff31        blx     r1

c044240c <pegmatite_smp_jump_table>:
c044240c:       00000000        .word   0x00000000

c0442410 <pegmatite_smp_jump_size>:
c0442410:       00000018        .word   0x00000018
```

size = 0x18 < 0x20

④

map physical address 0xd1000000-0xd100001f to virtual address

⑤

static unsigned long *pegmatite_boot_addr;

sizeof(pegmatite_boot_addr[0]) * num_present_cpus() = sizeof(unsigned long) * 2

即分配2个unsigned long space(每个core一个)

这里就是动态分配了pegmatite_boot_addr[] array,array的大小是core number

对gemstone2是

pegmatite_boot_addr[0]

pegmatite_boot_addr[1]

对granite2是

pegmatite_boot_addr[0]

pegmatite_boot_addr[1]

pegmatite_boot_addr[2]

pegmatite_boot_addr[3]

⑥

virt_to_phys(pegmatite_boot_addr) 获得pegmatite_boot_addr[]的physical address,然

后把该值赋值给pegmatite_smp_jump_table。

pegmatite_smp_jump_table位于assemble code中

```
ENTRY(pegmatite_smp_jump)
        ldr     r1, pegmatite_smp_jump_table
                (A)
        mrc     p15, 0, r0, c0, c0, 5   @ Read MPIDR
        (B)
        and     r0, r0, #0xf            @ Get CPU number in R0
                (C)
        ldr     r1, [r1, r0, LSL #2]    @ Read jump address for this cpu
        (D)
        blx     r1
                                        (E)

        .globl  pegmatite_smp_jump_table
pegmatite_smp_jump_table:
        .long   0x0
ENDPROC(pegmatite_smp_jump)
        .globl pegmatite_smp_jump_size
pegmatite_smp_jump_size:
        .long   . - pegmatite_smp_jump
```

这里之所以是physical address，是因为secondary code刚起来时还没有启用MMU。

(A)

获得pegmatite_smp_jump_table的值，也就是上面动态分配的pegmatite_boot_addr[]的首地址

(B)

读取secondary code的MPIDR register (cp15's c0's 5th register) in r0 register

从这行code可以看出每个core有自己独立的cp15 co-processor。这里读取的是secondary的cp15

的MPIDR register

(C)

取MPIDR register的低4位，从comment看好像是core的编号。即core 0应该编号为0,core 1编号为1，依次类推。

(D)

r1 = pegmatite_boot_addr[core_id]

pegmatite_boot_addr[]中应该fill 某种function address。现在还看不出来。

在run pegmatite_smp_prepare_cpus()时，pegmatite_boot_addr[]还没有fill有意义的function address。其实它是在smp_init()中填写的(step two of smp initialization II)。

in drivers/platform/pegmatite/smp/platsmp.c

pegmatite_boot_cpus_to() will fill the function table (pegmatite_boot_addr[]).

```
    for_each_cpu(cpu, cpus)
        writel(address, &pegmatite_boot_addr[cpu]);
    __cpuc_clean_dcache_area((void *)pegmatite_boot_addr, sizeof(pegmatite_boot_addr[0]) * num_present_cpus());
```

这里address = virt_to_phys(&pegmatite_secondary_startup)

比如pegmatite_boot_addr[1] = physical address of pegmatite_secondary_startup

这样对secondary core而言，blx r1将跳转到pegmatite_secondary_startup() in drivers/platform/pegmatite/smp/headsmp.S

(E)

跳转到某种function去执行。

⑦

把pegmatite_smp_jump() (定义在assemble code中)复制到squ_addr中。

⑧

内存屏障，同步一下可能乱序(out of order)的instruction。