in arch/arm/kernel/setup.c

```
1.  void __init setup_arch(char **cmdline_p)
2.  {
3.          const struct machine_desc *mdesc;
4.
5.          setup_processor();
6.          mdesc = setup_machine_fdt(__atags_pointer);
7.          if (!mdesc)
8.                  mdesc = setup_machine_tags(__atags_pointer, __machine_arch_type);
9.          machine_desc = mdesc;
10.         machine_name = mdesc->name;
11.
12. ......
13. }
```

arch相关code的第一步就是确定ARM core的arch？ARMv5, ARMv6 or ARMv7?

```
1.  static void __init setup_processor(void)
2.  {
3.          struct proc_info_list *list;
4.
5.          /*
6.           * locate processor in the list of supported processor
7.           * types.  The linker builds this table for us from the
8.           * entries in arch/arm/mm/proc-*.S
9.           */
10.         list = lookup_processor_type(read_cpuid_id());
11.         if (!list) {
12.                 pr_err("CPU configuration botched (ID %08x), unable to continue.\
    n",
13.                        read_cpuid_id());
14.                 while (1);
15.         }
16.
17.         cpu_name = list->cpu_name;
18.         __cpu_architecture = __get_cpu_architecture();
19.
20.  #ifdef MULTI_CPU
21.         processor = *list->proc;
22.  #endif
23.  #ifdef MULTI_TLB
24.         cpu_tlb = *list->tlb;
25.  #endif
26.  #ifdef MULTI_USER
27.         cpu_user = *list->user;
28.  #endif
29.  #ifdef MULTI_CACHE
30.         cpu_cache = *list->cache;
```

```
#endif

        pr_info("CPU: %s [%08x] revision %d (ARMv%s), cr=%08lx\n",
                cpu_name, read_cpuid_id(), read_cpuid_id() & 15,
                proc_arch[cpu_architecture()], get_cr());

        snprintf(init_utsname()->machine, __NEW_UTS_LEN + 1, "%s%c",
                 list->arch_name, ENDIANNESS);
        snprintf(elf_platform, ELF_PLATFORM_SIZE, "%s%c",
                 list->elf_name, ENDIANNESS);
        elf_hwcap = list->elf_hwcap;

        cpuid_init_hwcaps();

#ifndef CONFIG_ARM_THUMB
        elf_hwcap &= ~(HWCAP_THUMB | HWCAP_IDIVT);
#endif
#ifdef CONFIG_MMU
        init_default_cache_policy(list->__cpu_mm_mmu_flags);
#endif
        erratum_a15_798181_init();

        elf_hwcap_fixup();

        cacheid_init();
        cpu_init();
}
```

```
1.  static int __get_cpu_architecture(void)
2.  {
3.          int cpu_arch;
4.
5.          if ((read_cpuid_id() & 0x0008f000) == 0) {
6.                  cpu_arch = CPU_ARCH_UNKNOWN;
7.          } else if ((read_cpuid_id() & 0x0008f000) == 0x00007000) {
8.                  cpu_arch = (read_cpuid_id() & (1 << 23)) ? CPU_ARCH_ARMv4T : CPU_
    ARCH_ARMv3;
9.          } else if ((read_cpuid_id() & 0x00080000) == 0x00000000) {
10.                 cpu_arch = (read_cpuid_id() >> 16) & 7;
11.                 if (cpu_arch)
12.                         cpu_arch += CPU_ARCH_ARMv3;
13.         } else if ((read_cpuid_id() & 0x000f0000) == 0x000f0000) {
14.                 unsigned int mmfr0;
15.
16.                 /* Revised CPUID format. Read the Memory Model Feature
17.                  * Register 0 and check for VMSAv7 or PMSAv7 */
18.                 asm("mrc        p15, 0, %0, c0, c1, 4"
19.                     : "=r" (mmfr0));
20.                 if ((mmfr0 & 0x0000000f) >= 0x00000003 ||
21.                     (mmfr0 & 0x000000f0) >= 0x00000030)
22.                         cpu_arch = CPU_ARCH_ARMv7;
23.                 else if ((mmfr0 & 0x0000000f) == 0x00000002 ||
24.                          (mmfr0 & 0x000000f0) == 0x00000020)
25.                         cpu_arch = CPU_ARCH_ARMv6;
26.                 else
27.                         cpu_arch = CPU_ARCH_UNKNOWN;
28.         } else
29.                 cpu_arch = CPU_ARCH_UNKNOWN;
30.
```

```
31.         return cpu_arch;
32.  }
```

#define CPUID_ID 0

```
1.  static inline unsigned int __attribute_const__ read_cpuid_id(void)
2.  {
3.          return read_cpuid(CPUID_ID);
4.  }
```

```
1.  #define read_cpuid(reg)                                                 \
2.          ({                                                              \
3.                  unsigned int __val;                                     \
4.                  asm("mrc        p15, 0, %0, c0, c0, " __stringify(reg)  \
5.                      : "=r" (__val)                                      \
6.                      :                                                   \
7.                      : "cc");                                            \
8.                  __val;                                                  \
9.          })
```

mrc p15, 0, val, c0, c0, CPUID_ID

mrc p15, 0, r0, c0, c0, 0

将c0,0的值读到r0中。

c0,0 ==> MIDR, Main ID Register

val中为cpuid的值。

XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX

XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,**XXXX**

Revision

XXXX,XXXX,XXXX,XXXX,**XXXX,XXXX,XXXX**,XXXX

Primary part number

XXXX,XXXX,XXXX,**XXXX**,XXXX,XXXX,XXXX,XXXX

Architecture

Bits[19:16]	Architecture
0x1	ARMv4
0x2	ARMv4T
0x3	ARMv5(obsolete)
0x4	ARMv5T
0x5	ARMv5TE
0x6	ARMv5TEJ
0x7	ARMv6
0xF	Defined by CPUID scheme

XXXX,XXXX,**XXXX**,XXXX,XXXX,XXXX,XXXX,XXXX

Variant

XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX

Implementer

```
asm("mrc        p15, 0, %0, c0, c1, 4"
    : "=r" (mmfr0));
```

mrc p15, 0, r0, c0, c1, 4

c0,c1,4 ==> ID_MMFR0, Memory Model Feature Register 0

c0,c1,5 ==> ID_MMFR1, Memory Model Feature Register 1

c0,c1,6 ==> ID_MMFR2, Memory Model Feature Register 2

c0,c1,7 ==> ID_MMFR3, Memory Model Feature Register 3