u-boot在armv7/armv7m/armv8的入口分别在
arch/arm/cpu/armv7/start.S
arch/arm/cpu/armv7m/start.S
arch/arm/cpu/armv8/start.S

对armv7 core的大致流程如下

arch/arm/cpu/armv7/start.S

```
/*
 * Setup vector:
 * (OMAP4 spl TEXT_BASE is not 32 byte aligned.
 * Continue to use ROM code vector only in OMAP4 spl)
 */
#if !(defined(CONFIG_OMAP44XX) && defined(CONFIG_SPL_BUILD))
    /* Set V=0 in CP15 SCTLR register - for VBAR to point to vector */
    mrc p15, 0, r0, c1, c0, 0   @ Read CP15 SCTLR Register
    bic r0, #CR_V       @ V = 0
    mcr p15, 0, r0, c1, c0, 0   @ Write CP15 SCTLR Register

    /* Set vector address in CP15 VBAR register */
    ldr r0, =_start
    mcr p15, 0, r0, c12, c0, 0  @Set VBAR
#endif

    /* the mask ROM code should have PLL and others stable */
#ifndef CONFIG_SKIP_LOWLEVEL_INIT
    bl  cpu_init_cp15
    bl  cpu_init_crit
#endif

    bl  _main
```

- 指定vector table

arch	file
armv7	arch/arm/lib/vectors.S
armv7m	arch/arm/lib/vectors_m.S
armv8	arch/arm/cpu/armv8/start.S

- init cp15 register
  cpu_init_cp15()
  arch/arm/cpu/armv7/start.S
  u-boot提供
- board specific init

```
ENTRY(cpu_init_crit)
    /*
     * Jump to board specific initialization...
     * The Mask ROM will have already initialized
     * basic memory. Go here to bump up clock rate and handle
     * wake up conditions.
     */
    b   lowlevel_init       @ go setup pll,mux,memory
ENDPROC(cpu_init_crit)
```

这里的lowlevel_init()就是板子移植的工作

```
lowlevel_init():
    - purpose: essential init to permit execution to reach board_init_f()
    - no global_data or BSS
    - there is no stack (ARMv7 may have one but it will soon be removed)
    - must not set up SDRAM or use console
    - must only do the bare minimum to allow execution to continue to
        board_init_f()
    - this is almost never needed
    - return normally from this function
```

在pegmatite中

in arch/arm/cpu/armv7/pegmatite/smp_init.S

```
.globl lowlevel_init
    .type   lowlevel_init, %function
lowlevel_init:
    @ Save link register in r3
    mov r3, lr

    mrrc    p15, 1, r0, r1, c15 @ Read CPUECTRL register
    orr r0, r0, #(1<<6)         @ Turn on SMP
    mcrr    p15, 1, r0, r1, c15 @ Write CPUECTRL register

    mov pc, r3          @ return
```

- jump to main()

```
bl  _main
```

in arch/arm/lib/crt0.S

```
main()
{
    board_init_f();
    relocate_code();
    board_init_r();
}
```

根据u-boot/README

```
board_init_f():
    - purpose: set up the machine ready for running board_init_r():
      i.e. SDRAM and serial UART
    - global_data is available
    - stack is in SRAM
    - BSS is not available, so you cannot use global/static variables,
      only stack variables and global_data
```

```
board_init_r():
    - purpose: main execution, common code
    - global_data is available
    - SDRAM is available
    - BSS is available, all static/global variables can be used
    - execution eventually continues to main_loop()
```

在u-boot 2015.01中

remove arch/arm/lib/board.c(没有了arch specific board initialization)

arm arch已经不支持arch specific board initialization,有些arch还支持

in pegmatite.h

```
#define CONFIG_SYS_GENERIC_BOARD 1
#define CONFIG_BOARD_EARLY_INIT_F 1
#define CONFIG_BOARD_LATE_INIT  1
```

```
- CONFIG_SYS_GENERIC_BOARD
    This selects the architecture-generic board system instead of the
    architecture-specific board files. It is intended to move boards
    to this new framework over time. Defining this will disable the
    arch/foo/lib/board.c file and use common/board_f.c and
    common/board_r.c instead. To use this option your architecture
    must support it (i.e. must select HAVE_GENERIC_BOARD in arch/Kconfig).
    If you find problems enabling this option on your board please report
    the problem and send patches!
```

```
- CONFIG_BOARD_EARLY_INIT_F: Call board_early_init_f()
- CONFIG_BOARD_EARLY_INIT_R: Call board_early_init_r()
```