atags是device tree出现以前,u-boot给kernel传递信息的方式。（atags和device tree两种方式是互斥的）

Creating atags is in u-boot/arch/arm/lib/bootm.c

```
static void boot_prep_linux(bootm_headers_t *images)
{
        char *commandline = getenv("bootargs");

        if (IMAGE_ENABLE_OF_LIBFDT && images->ft_len) {
#ifdef CONFIG_OF_LIBFDT
                debug("using: FDT\n");
                if (image_setup_linux(images)) {
                        printf("FDT creation failed! hanging...");
                        hang();
                }
#endif
        } else if (BOOTM_ENABLE_TAGS) {
                debug("using: ATAGS\n");
                setup_start_tag(gd->bd);
                if (BOOTM_ENABLE_SERIAL_TAG)
                        setup_serial_tag(¶ms);
                if (BOOTM_ENABLE_CMDLINE_TAG)
                        setup_commandline_tag(gd->bd, commandline);
                if (BOOTM_ENABLE_REVISION_TAG)
                        setup_revision_tag(¶ms);
                if (BOOTM_ENABLE_MEMORY_TAGS)
                        setup_memory_tags(gd->bd);
                if (BOOTM_ENABLE_INITRD_TAG) {
                        if (images->rd_start && images->rd_end) {
                                setup_initrd_tag(gd->bd, images->rd_start,
                                                 images->rd_end);
                        }
                }
                setup_board_tags(¶ms);
                setup_end_tag(gd->bd);
        } else {
                printf("FDT and ATAGS support not compiled in - hanging\n");
                hang();
        }
        do_nonsec_virt_switch();
}
```

in u-boot/arch/arm/include/asm/bootm.h

```
#if defined(CONFIG_SETUP_MEMORY_TAGS) || \
                defined(CONFIG_CMDLINE_TAG) || \
                defined(CONFIG_INITRD_TAG) || \
                defined(CONFIG_SERIAL_TAG) || \
                defined(CONFIG_REVISION_TAG)
# define BOOTM_ENABLE_TAGS              1
#else
# define BOOTM_ENABLE_TAGS              0
#endif
```

控制TAGS生成与否。

生成的atags存放在gd->bd->bi_boot_params中。

gd是gd_t global variable.

in u-boot/include/asm-generic/global_data.h

```
/*
 * The following data structure is placed in some memory which is
 * available very early after boot (like DPRAM on MPC8xx/MPC82xx, or
 * some locked parts of the data cache) to allow for a minimum set of
 * global variables during system initialization (until we have set
 * up the memory controller so that we can use RAM).
 *
 * Keep it *SMALL* and remember to set GENERATED_GBL_DATA_SIZE > sizeof(gd_t)
 *
 * Each architecture has its own private fields. For now all are private
 */

#ifndef __ASSEMBLY__
typedef struct global_data {
        bd_t *bd;
        unsigned long flags;
        unsigned int baudrate;
        unsigned long cpu_clk;  /* CPU clock in Hz!             */
        unsigned long bus_clk;
        /* We cannot bracket this with CONFIG_PCI due to mpc5xxx */
        unsigned long pci_clk;
        unsigned long mem_clk;
#if defined(CONFIG_LCD) || defined(CONFIG_VIDEO) || 1
        unsigned long fb_base;  /* Base address of framebuffer mem */
#endif
#if defined(CONFIG_POST) || defined(CONFIG_LOGBUFFER)
        unsigned long post_log_word;  /* Record POST activities */
        unsigned long post_log_res; /* success of POST test */
        unsigned long post_init_f_time;  /* When post_init_f started */
#endif
#ifdef CONFIG_BOARD_TYPES
        unsigned long board_type;
#endif
        unsigned long have_console;     /* serial_init() was called */
#ifdef CONFIG_PRE_CONSOLE_BUFFER
        unsigned long precon_buf_idx;   /* Pre-Console buffer index */
#endif
#ifdef CONFIG_MODEM_SUPPORT
        unsigned long do_mdm_init;
        unsigned long be_quiet;
#endif
        unsigned long env_addr; /* Address  of Environment struct */
        unsigned long env_valid;        /* Checksum of Environment valid? */

        unsigned long ram_top;  /* Top address of RAM used by U-Boot */

        unsigned long relocaddr;        /* Start address of U-Boot in RAM */
        phys_size_t ram_size;   /* RAM size */
        unsigned long mon_len;  /* monitor len */
        unsigned long irq_sp;           /* irq stack pointer */
        unsigned long start_addr_sp;    /* start_addr_stackpointer */
        unsigned long reloc_off;
        struct global_data *new_gd;     /* relocated global data */
        const void *fdt_blob;   /* Our device tree, NULL if none */
        void *new_fdt;          /* Relocated FDT */
        unsigned long fdt_size; /* Space reserved for relocated FDT */
        void **jt;              /* jump table */
        char env_buf[32];       /* buffer for getenv() before reloc. */
#ifdef CONFIG_TRACE
        void            *trace_buff;    /* The trace buffer */
#endif
#if defined(CONFIG_SYS_I2C)
        int             cur_i2c_bus;    /* current used i2c bus */
#endif
        unsigned long timebase_h;
        unsigned long timebase_l;
        struct arch_global_data arch;   /* architecture-specific data */
} gd_t;
#endif
```

---

kernel handles atag in arch/arm/kernel/setup.c

```
void __init setup_arch(char **cmdline_p)
{
        const struct machine_desc *mdesc;

        setup_processor();
        mdesc = setup_machine_fdt(__atags_pointer);
        if (!mdesc)
                                        ①
                mdesc = setup_machine_tags(__atags_pointer, __machine_arch_type);
   ②
        machine_desc = mdesc;
        machine_name = mdesc->name;
......
}
```

①

只有在没有enable device tree的情况下才会parse atag

②

__atags_pointer要么指向device tree blob,要么指向atags。

in arch/arm/kernel/atags_parse.c

```
const struct machine_desc * __init
setup_machine_tags(phys_addr_t __atags_pointer, unsigned int machine_nr)
{
        struct tag *tags = (struct tag *)&default_tags;
        const struct machine_desc *mdesc = NULL, *p;
        char *from = default_command_line;

        default_tags.mem.start = PHYS_OFFSET;

        /*
         * locate machine in the list of supported machines.
         */
        for_each_machine_desc(p)
                if (machine_nr == p->nr) {
                        printk("Machine: %s\n", p->name);
                        mdesc = p;
                        break;
                }

        if (!mdesc) {
                early_print("\nError: unrecognized/unsupported machine ID"
                            " (r1 = 0x%08x).\n\n", machine_nr);
                dump_machine_table(); /* does not return */
        }

        if (__atags_pointer)
                tags = phys_to_virt(__atags_pointer);
        else if (mdesc->atag_offset)
                tags = (void *)(PAGE_OFFSET + mdesc->atag_offset);

#if defined(CONFIG_DEPRECATED_PARAM_STRUCT)
        /*
         * If we have the old style parameters, convert them to
         * a tag list.
         */
        if (tags->hdr.tag != ATAG_CORE)
                convert_to_tag_list(tags);
#endif
        if (tags->hdr.tag != ATAG_CORE) {
                early_print("Warning: Neither atags nor dtb found\n");
                tags = (struct tag *)&default_tags;
        }

        if (mdesc->fixup)
                mdesc->fixup(tags, &from);

        if (tags->hdr.tag == ATAG_CORE) {
                if (memblock_phys_mem_size())
                        squash_mem_tags(tags);
                save_atags(tags);
                parse_tags(tags);
        }

        /* parse_early_param needs a boot_command_line */
        strlcpy(boot_command_line, from, COMMAND_LINE_SIZE);

        return mdesc;
}
```