in arm arch,在有了arch/arm/setup.c中的early_print()以后，实在不知道为什么还需要arch/arm/kernel/early_printk.c?!

in arch/arm/setup.c

```
void __init early_print(const char *str, ...)
{
        extern void printascii(const char *);
        char buf[256];
        va_list ap;

        va_start(ap, str);
        vsnprintf(buf, sizeof(buf), str, ap);
        va_end(ap);

#ifdef CONFIG_DEBUG_LL
        printascii(buf);
#endif
        printk("%s", buf);
}
```

early_print()直接调用汇编级的printascii() (in arch/arm/kernel/debug.S)

而early_printk feature

```
extern void printch(int);


static void early_write(const char *s, unsigned n)

{

        while (n-- > 0) {

                if (*s == '\n')

                        printch('\r');

                printch(*s);

                s++;

        }

}


static void early_console_write(struct console *con, const char *s, unsigned n)

{

        early_write(s, n);

}


static struct console early_console_dev = {

        .name =         "earlycon",

        .write =    early_console_write,

        .flags =    CON_PRINTBUFFER | CON_BOOT,

        .index =    -1,

};


static int __init setup_early_printk(char *buf)
```

```
{
        early_console = &early_console_dev;
        register_console(&early_console_dev);
        return 0;
}

early_param("earlyprintk", setup_early_printk);
```

early_printk feature 本质上也是调用arch/arm/kernel/debug.S中的汇编级函数printch(),只不过它register一个boot console,这样可以printk()而不是early_print()了。两者共通的是都依赖于CONFIG_DEBUG_LL。

```
obj-$(CONFIG_DEBUG_LL)  += debug.o
```

如果没有配置CONFIG_DEBUG_LL，则都无法工作。

另外由于在kernel/printk/printk.c中

```
#ifdef CONFIG_EARLY_PRINTK
struct console *early_console;

void early_vprintk(const char *fmt, va_list ap)
{
        if (early_console) {
                char buf[512];
                int n = vscnprintf(buf, sizeof(buf), fmt, ap);
                early_console->write(early_console, buf, n);
        }
}

asmlinkage __visible void early_printk(const char *fmt, ...)
{
        va_list ap;

        va_start(ap, fmt);
        early_vprintk(fmt, ap);
        va_end(ap);
}
#endif
```

这样在"early_print"阶段，也就是start_kernel()的console_init()之前，输出log可以由三种形式：

1. early_print()
2. printk
3. early_printk()

其实这三者本质上都一样，都是通过配置CONFIG_DEBUG_LL开直接操作UART输出。

in Documentation/kernel-parameters.txt

```
earlyprintk=	[X86,SH,BLACKFIN,ARM,M68k]
		earlyprintk=vga
		earlyprintk=efi
		earlyprintk=xen
		earlyprintk=serial[,ttySn[,baudrate]]
		earlyprintk=serial[,0x...[,baudrate]]
		earlyprintk=ttySn[,baudrate]
		earlyprintk=dbgp[debugController#]

		earlyprintk is useful when the kernel crashes before
		the normal console is initialized. It is not enabled by
		default because it has some cosmetic problems.

		Append ",keep" to not disable it when the real console
		takes over.

		Only one of vga, efi, serial, or usb debug port can
		be used at a time.

		Currently only ttyS0 and ttyS1 may be specified by
		name.  Other I/O ports may be explicitly specified
		on some architectures (x86 and arm at least) by
		replacing ttySn with an I/O port address, like this:
			earlyprintk=serial,0x1008,115200
		You can find the port for a given device in
```

/proc/tty/driver/serial:

```
        2: uart:ST16650V2 port:00001008 irq:18 ...
```

Interaction with the standard serial driver is not very good.

The VGA and EFI output is eventually overwritten by the real console.

The xen output can only be used by Xen PV guests.