接下来，就可以使用 GDB 的单步调试命令单步调试 head.S 了。

3.2 ftrace

ftrace 最早出现在 Linux 2.6.27 内核中，其设计目标简单，基于静态代码插桩 (stub) 技术，不需要用户通过额外的编程来定义 trace 行为。静态代码插桩技术比较可靠，不会因为用户使用不当而导致内核崩溃。ftrace 的名字源于 function trace，利用 GCC 的 profile 特性在所有函数入口处添加一段插桩代码，ftrace 重载这段代码来实现 trace 功能。GCC 的 -pg 选项会在每个函数入口处加入 mcount 的调用代码，原本 mcount 有 libc 实现，而内核不会链接 libc 库，因此 ftrace 编写了自己的 mcount stub 函数。

在使用 ftrace 之前，需要确保内核编译了配置选项。

```
CONFIG_FTRACE=y
CONFIG_HAVE_FUNCTION_TRACER=y
CONFIG_HAVE_FUNCTION_GRAPH_TRACER=y
CONFIG_HAVE_DYNAMIC_FTRACE=y
CONFIG_FUNCTION_TRACER=y
CONFIG_IRQSOFF_TRACER=y
CONFIG_SCHED_TRACER=y
CONFIG_ENABLE_DEFAULT_TRACERS=y
CONFIG_FTRACE_SYSCALLS=y
CONFIG_PREEMPT_TRACER=y
```

ftrace 的相关配置选项比较多，针对不同的跟踪器有各自对应的配置选项。ftrace 通过 debugfs 文件系统向用户空间提供访问接口，因此需要在系统启动时挂载 debugfs，可以修改系统的/etc/fstab 文件或手动挂载。

```
mount -t debugfs debugfs /sys/kernel/debug
```

在 /sys/kernel/debug/trace 目录下提供了各种跟踪器 (tracer) 和事件 (event)，一些常用的选项如下。

- ❑ available_tracers: 列出当前系统支持的跟踪器。
- ❑ available_events: 列出当前系统支持的事件。
- ❑ current_tracer: 设置和显示当前正在使用的跟踪器。使用 echo 命令把跟踪器的名字写入该文件，即可切换不同的跟踪器。默认为 nop，即不做任何跟踪操作。
- ❑ trace: 读取跟踪信息。通过 cat 命令查看 ftrace 记录下来的跟踪信息。
- ❑ tracing_on: 用于开始或暂停跟踪。
- ❑ trace_options: 设置 ftrace 的一些相关选项。

ftrace 当前包含多个跟踪器，方便用户跟踪不同类型的信息，如进程睡眠、唤醒、抢占、延迟的信息。查看 available_tracers 可以知道当前系统支持哪些跟踪器，如果系统支持的跟踪器上没有用户想要的，那就必须在配置内核时打开，然后重新编译内核。常用的 ftrace 跟踪器如表 3.1 所示。