卡通人物

描述已自动生成

**操作系统**

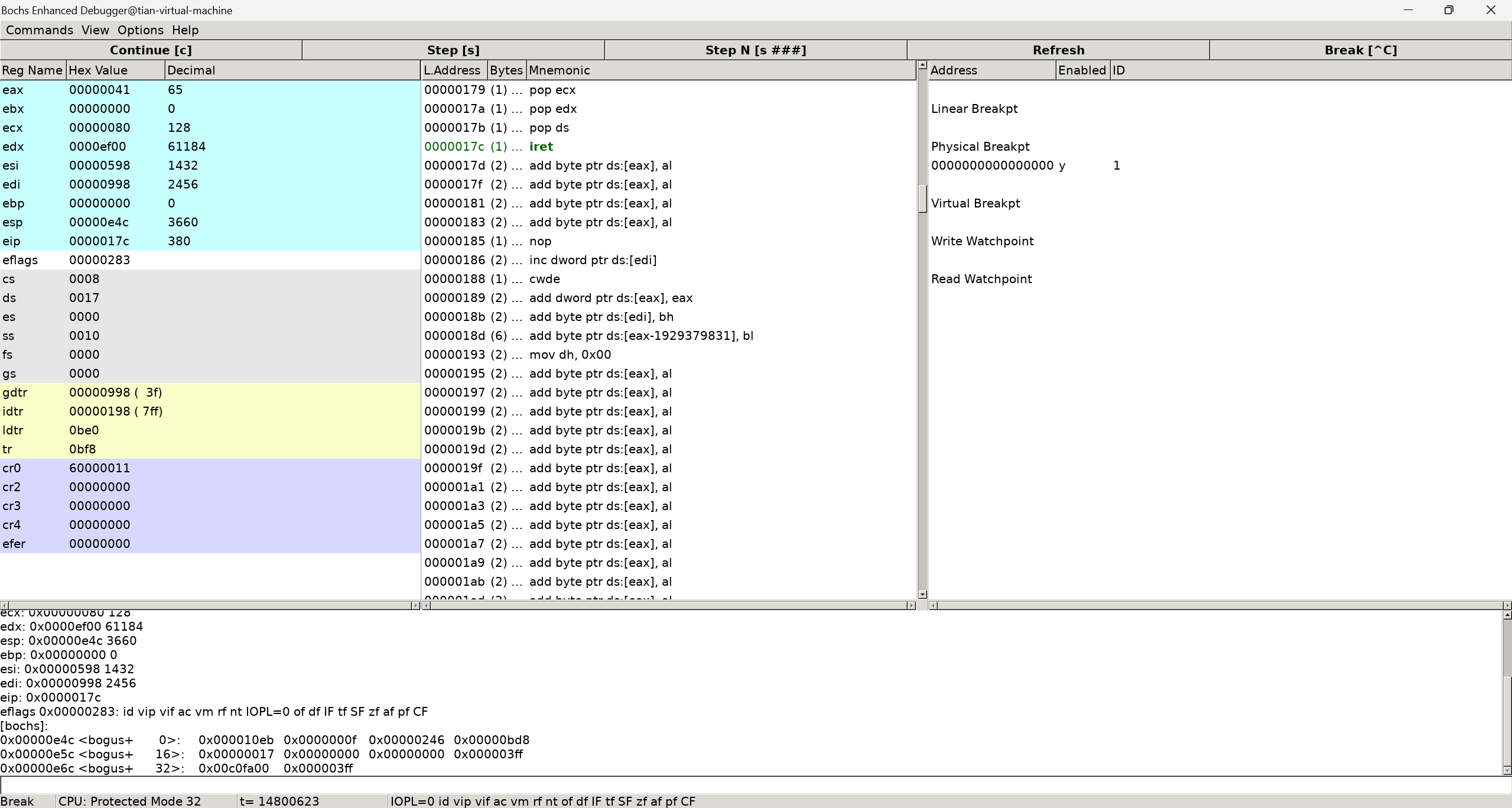
**课程实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | 调试分析 Linux 0.00 多任务切换 | | |
| 姓名 | 常添 | 院系 | 计算学部 |
| 班级 | 2203102 | 学号 | 2022111699 |
| 任课教师 | 郑铁然 | 指导教师 | 郑铁然 |
| 实验地点 | G709 | 实验时间 | 2024.11.24 |

****

**实验2：调试分析 Linux 0.00 多任务切换**

**任务 1：记录 system\_interrupt 执行完毕后，153 行 iret 的栈变化情况**

****

**执行 iret 前**：

* **栈指针位置**：SS:ESP = 0x10:0x0E4C。
* **内核栈内容**：

+---------------+

| 0x10EB | <- 用户态 EIP（返回地址）

+---------------+

| 0x0F | <- 用户态 CS（代码段选择子）

+---------------+

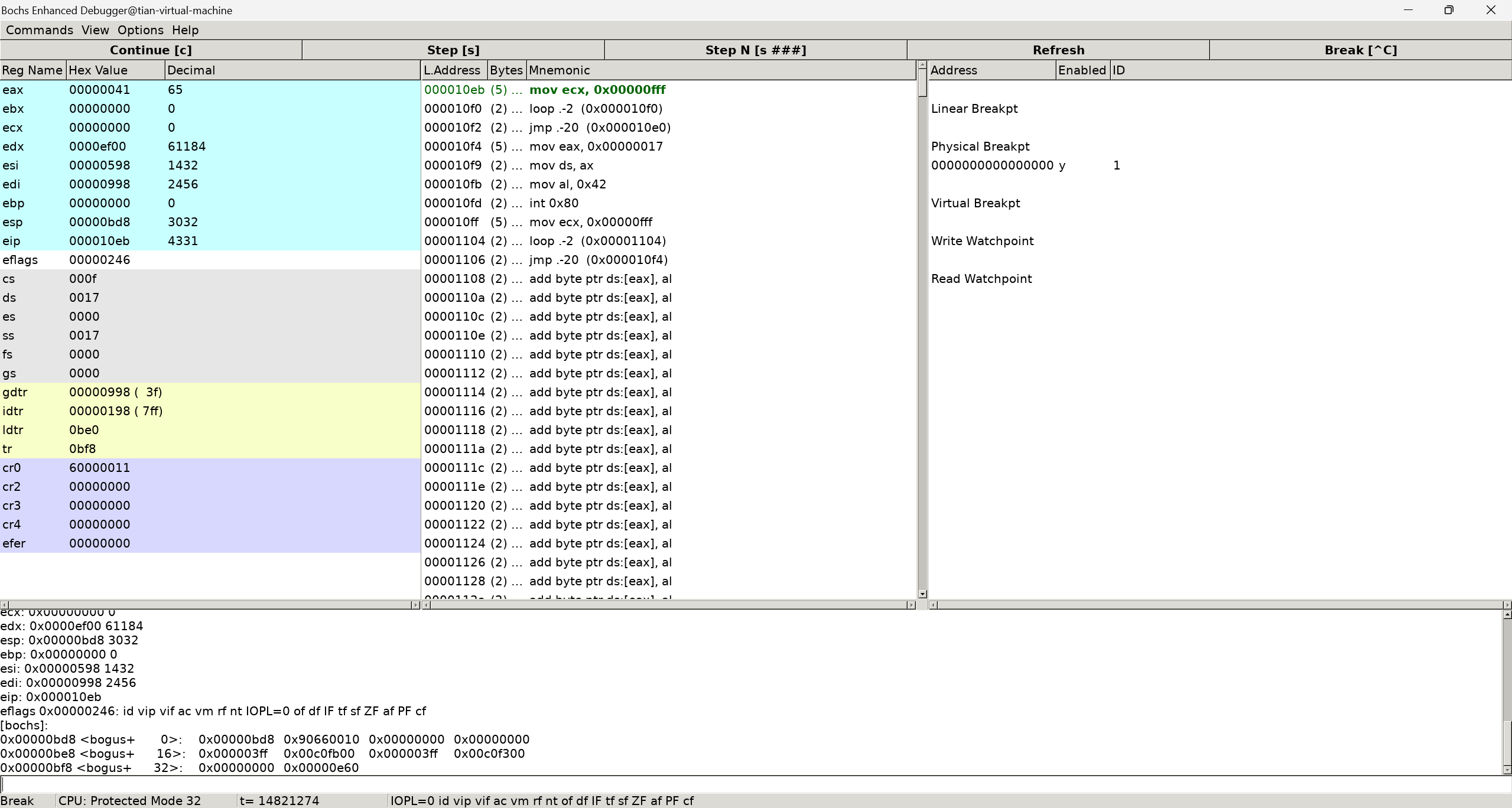
| 0x00246 | <- 用户态 EFLAGS（标志寄存器）

+---------------+

| 0x0BD8 | <- 用户态 ESP（用户栈顶地址）

+---------------+

| 0x17 | <- 用户态 SS（用户栈段选择子）

**执行 iret 后**：

* **栈指针位置**：SS:ESP = 0x17:0x0BD8。
* **栈内容切换**：
  + 内核栈内容被清空，切换到用户态栈。
  + iret 指令从内核栈中弹出 EIP、CS、EFLAGS，并恢复用户态栈 SS:ESP，返回到 CS:EIP = 0x0F:0x10EB 执行用户代码。

**任务 2：总结进入和退出 system\_interrupt 时的模式切换**

**进入 system\_interrupt（用户态 → 内核态）**

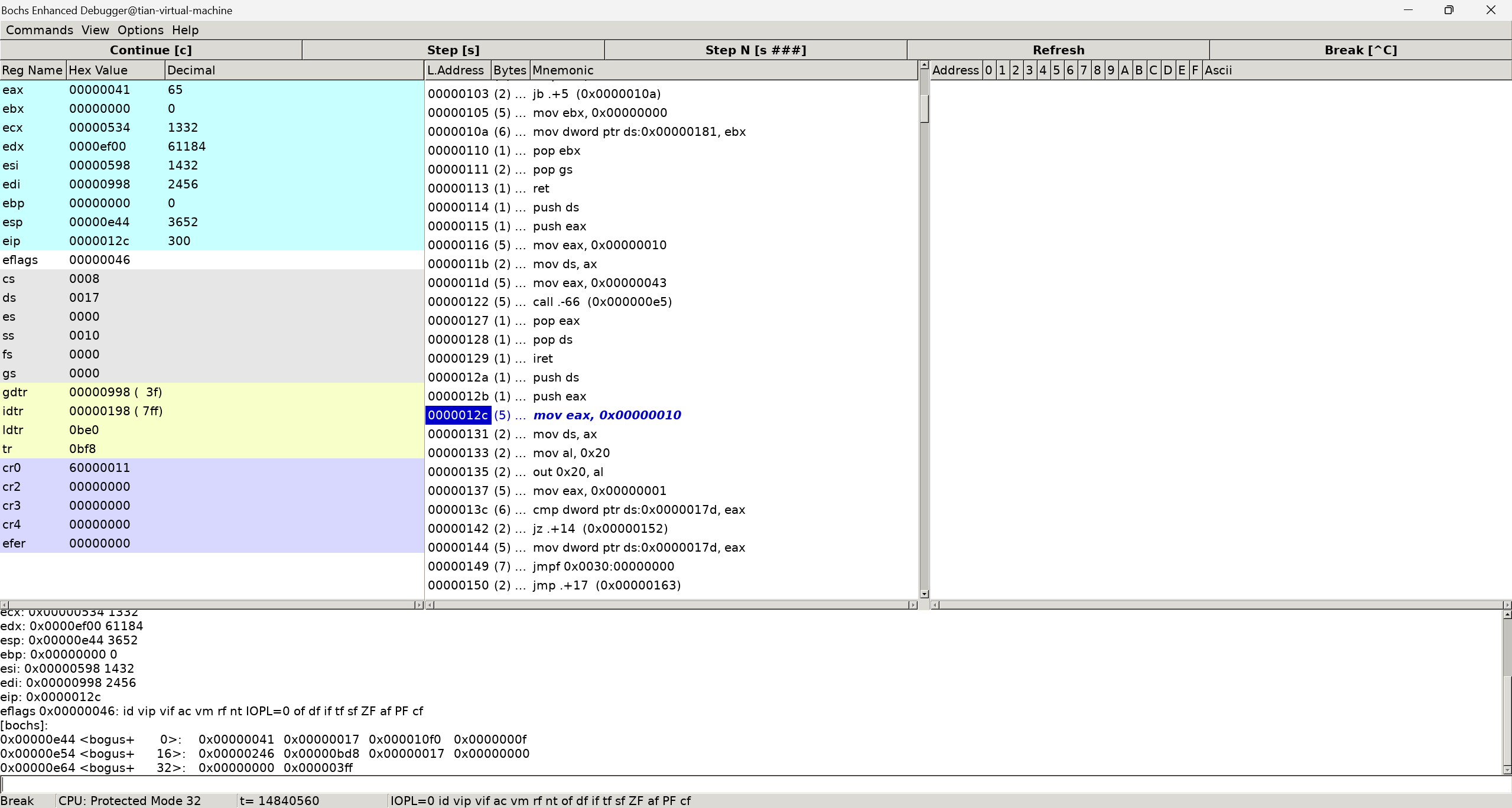
* **触发条件**：用户态程序通过 int 0x80 触发系统调用。
* **特权级变化**：从特权级 3（用户态）切换到特权级 0（内核态）。
* **栈切换**：
  1. **CPU 自动切换栈**：从 TSS 中加载内核态的 SS:ESP。
  2. **保存用户态状态到内核栈**：
     + 用户态的 SS:ESP（用户栈段选择子和栈指针）。
     + 用户态的 CS:EIP（返回地址）。
     + 用户态的 EFLAGS（标志寄存器）。
  3. 切换到内核栈，开始执行 system\_interrupt。

**退出 system\_interrupt（内核态 → 用户态）**

* **触发条件**：内核代码执行 iret 指令返回用户态。
* **特权级变化**：从特权级 0（内核态）切换到特权级 3（用户态）。
* **栈切换**：
  1. 从内核栈中弹出用户态的 CS:EIP、EFLAGS、SS:ESP。
  2. 恢复用户态的寄存器和栈状态，切换回用户栈，继续执行用户态代码。

**任务 3：记录从任务 0 切换到任务 1 的过程**

1. **时钟中断触发**：
   * 任务 0 正在执行，定时器中断触发 timer\_interrupt。
   * CPU 保存任务 0 的寄存器状态到内核栈。



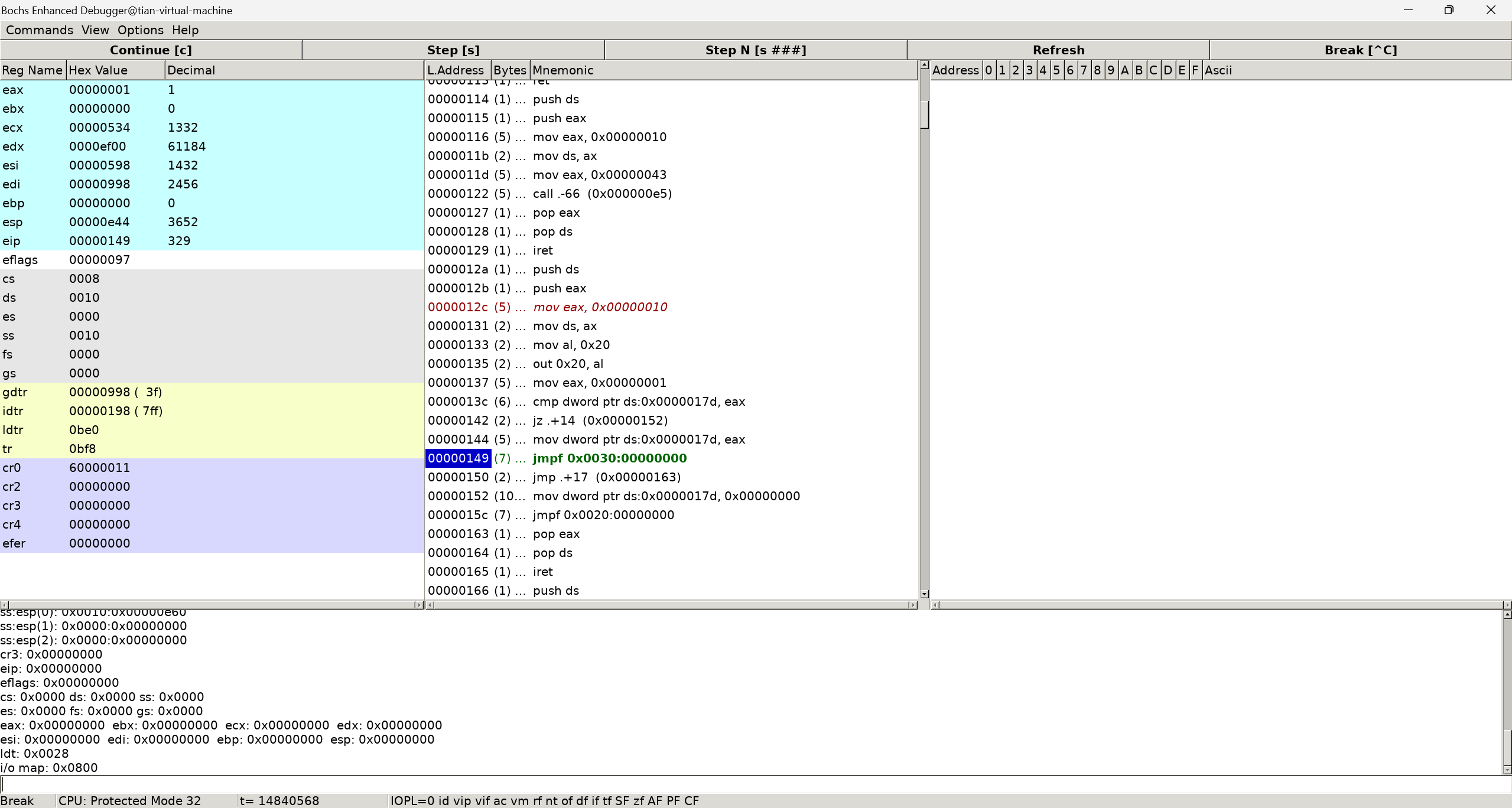
1. **保存任务 0 的状态**：
   * 使用 save\_tss 函数，将任务 0 的 SS:ESP、CS:EIP 和其他寄存器保存到任务 0 的 TSS。
   * 内核栈中保存寄存器内容。

图形用户界面, 文本

描述已自动生成



1. **选择任务 1**：
   * 调度器选择任务 1，将 GDT 中的 TSS 描述符更新为任务 1 的 TSS。
   * 使用 ljmp $0x30, $0 指令切换到任务 1 的 TSS。



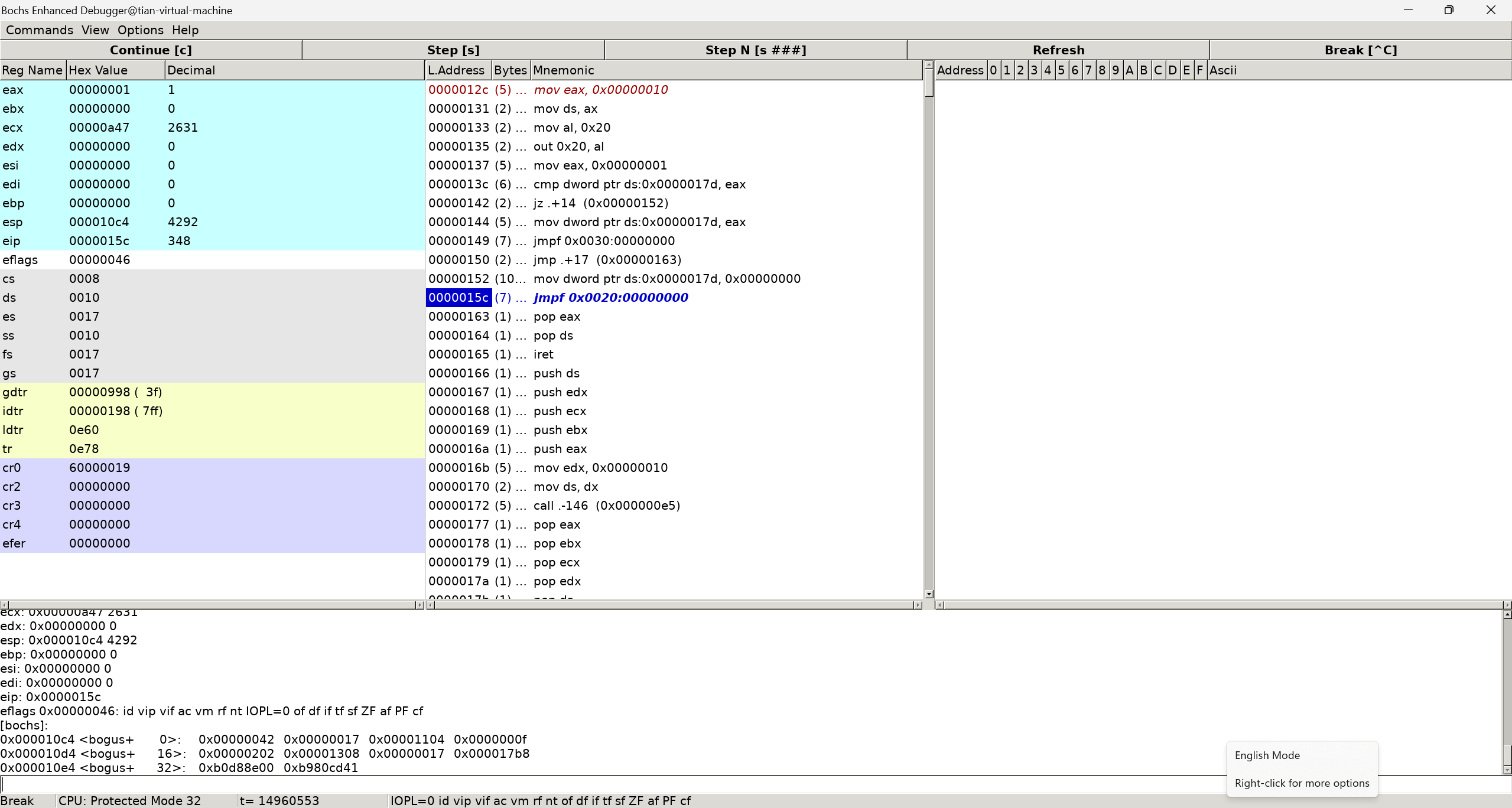
表格

低可信度描述已自动生成

1. **切换到任务 1**：
   * CPU 从任务 1 的 TSS 加载状态，包括 CS:EIP、SS:ESP 和其他寄存器。
   * CPU 开始执行任务 1 的用户代码 task1。

**任务 4：记录从任务 1 切换回任务 0 的过程**

1. **再次时钟中断触发**：
   * 任务 1 正在执行，定时器中断再次触发。
   * CPU 保存任务 1 的寄存器状态到内核栈。

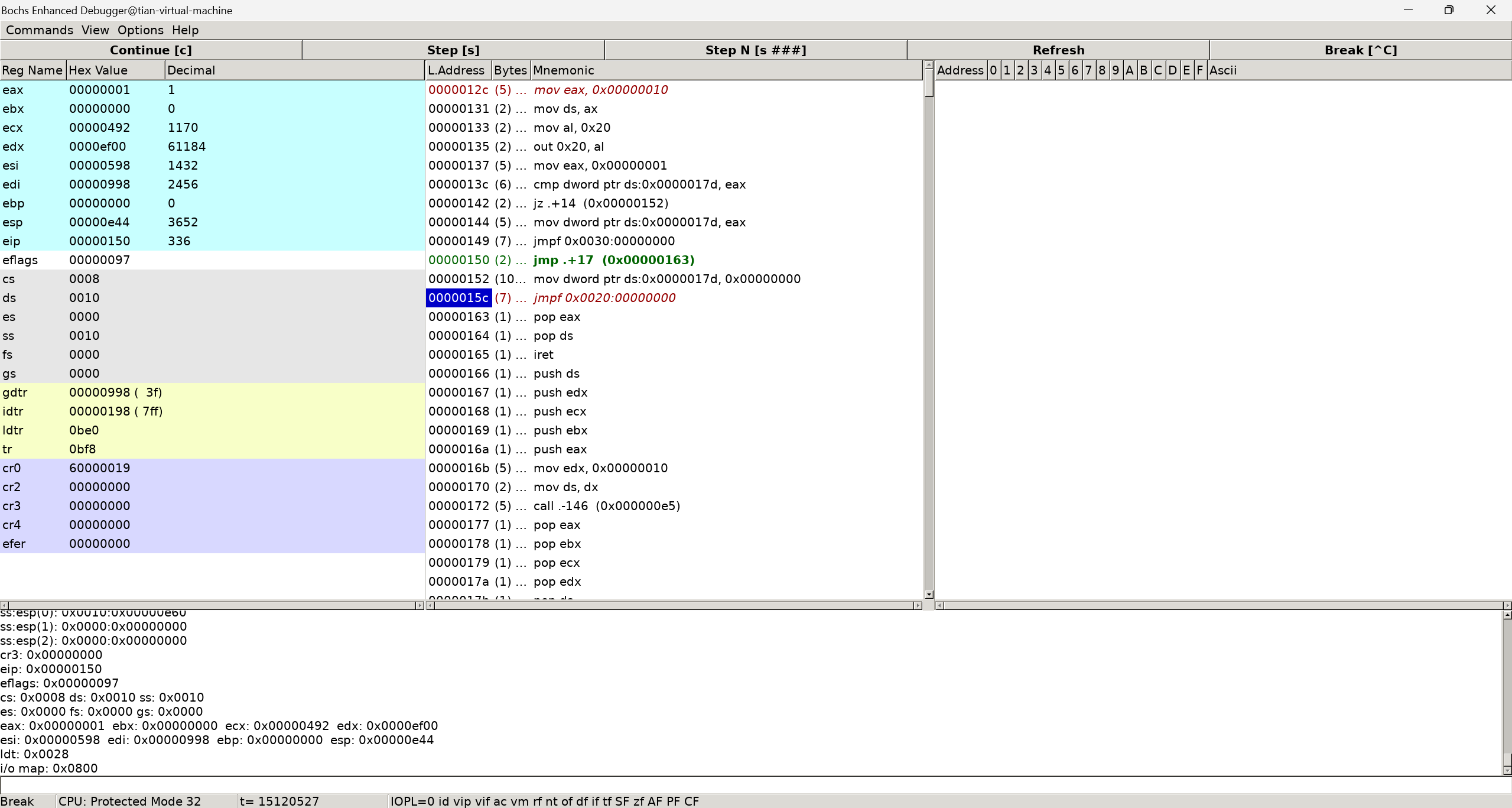


1. **保存任务 1 的状态**：
   * 使用 save\_tss 函数，将任务 1 的 SS:ESP、CS:EIP 和其他寄存器保存到任务 1 的 TSS。
   * 内核栈中保存寄存器内容。

表格

描述已自动生成

1. **选择任务 0**：
   * 调度器选择任务 0，将 GDT 中的 TSS 描述符更新为任务 0 的 TSS。
   * 使用 ljmp $0x20, $0 指令切换到任务 0 的 TSS。



1. **切换到任务 0**：
   * CPU 从任务 0 的 TSS 恢复状态，包括 CS:EIP = 0x08:0x0150，即任务 0 上次被中断的位置。
   * 恢复寄存器和栈状态，继续执行任务 0。

**任务 5：总结任务切换的过程**

1. **时钟中断触发**：
   * 定时器中断定期触发，进入 timer\_interrupt。
2. **保存当前任务状态**：
   * 当前任务的寄存器状态保存到内核栈。
   * 使用 save\_tss 函数，将当前任务的 SS:ESP、CS:EIP 和寄存器状态保存到对应的 TSS。
3. **调度器选择新任务**：
   * 调度器确定下一个任务。
   * 更新 GDT 中的 TSS 描述符指向新任务的 TSS。
4. **切换任务**：
   * 使用 ljmp 指令切换到新任务的 TSS。
   * CPU 从 TSS 加载新任务的 SS:ESP、CS:EIP 和其他寄存器状态。
5. **恢复并执行新任务**：
   * 恢复新任务的栈和寄存器状态。
   * 从新任务的 CS:EIP 开始执行。