# 嵌入式系统实验报告



|  |  |
| --- | --- |
| 实验名称： | Cortex-M 系统编程—异常处理 |
| 姓 名： | 王小龙 |
| 学 号： | 2020211502 |
| 学 院(系)： | 计算机学院 |
| 专 业： | 网络工程 |
| 指导教师： | 戴志涛、刘健培 |

2022年 12 月 1 日

# 实验目的

理解异常处理的过程，学会编写简单的异常处理程序。

# 实验环境

* ST-Link 仿真器
* KeiluVision5 MDK集成开发软件
* Window10 (64bit)

# 实验要求

Cortex-M 处理器在发生异常时，会进入异常处理程序。在调试环境下，可以通过调试器检查处理器状态，分析异常原因，但在生产环境下，往往无法连接调试器，一种做法是在异常处理程序中实现一个 coredump（核心转储）程序，将发生异常时程序的状态（内核寄存器值、堆栈、关键寄存器与数据结构等）输出到串口或者保存到文件，便于事后分析异常原因。

本实验要求实现一个简单的 coredump 程序，当程序发生异常时，可以在异常程序中将发生异常前 Cortex -M 处理器的核内寄存器的值用 printf 打印出来。

# 实验原理

当程序触发一个内存不对齐访问的异常（UNALIGN\_TRP），此异常会进入 HardFault\_Handler 处理，通过重写 HardFault\_Handler 以替换默认的汇编实现。

当处理器识别出异常时，会通过特定序列的行为来进行自动响应：

（1）压栈：将寄存器 R0～R3 和 R12 的内容、返回地址、PSR 和链接寄存器

LR 的内容压栈。

（2）在进行压栈操作的同时，处理器识别异常，将其对异常的识别结果作为PSR 中的异常号记录下来，然后使用该数字从异常向量表（vector table）中定位和拷贝对应异常处理程序的地址到程序计数器（PC）中。

（3）LR 寄存器装载除了最低的 4 位外全 1 的序列，最低 4 位记录指向异常的处理器模式（线程模式或者异常处理模式）。（注意：LR 不包含返回地址！LR中存储的值（通常为 EXC\_RETURN）作为特殊值由处理器识别，并用于在异常处理程序返回时恢复到之前的模式。）

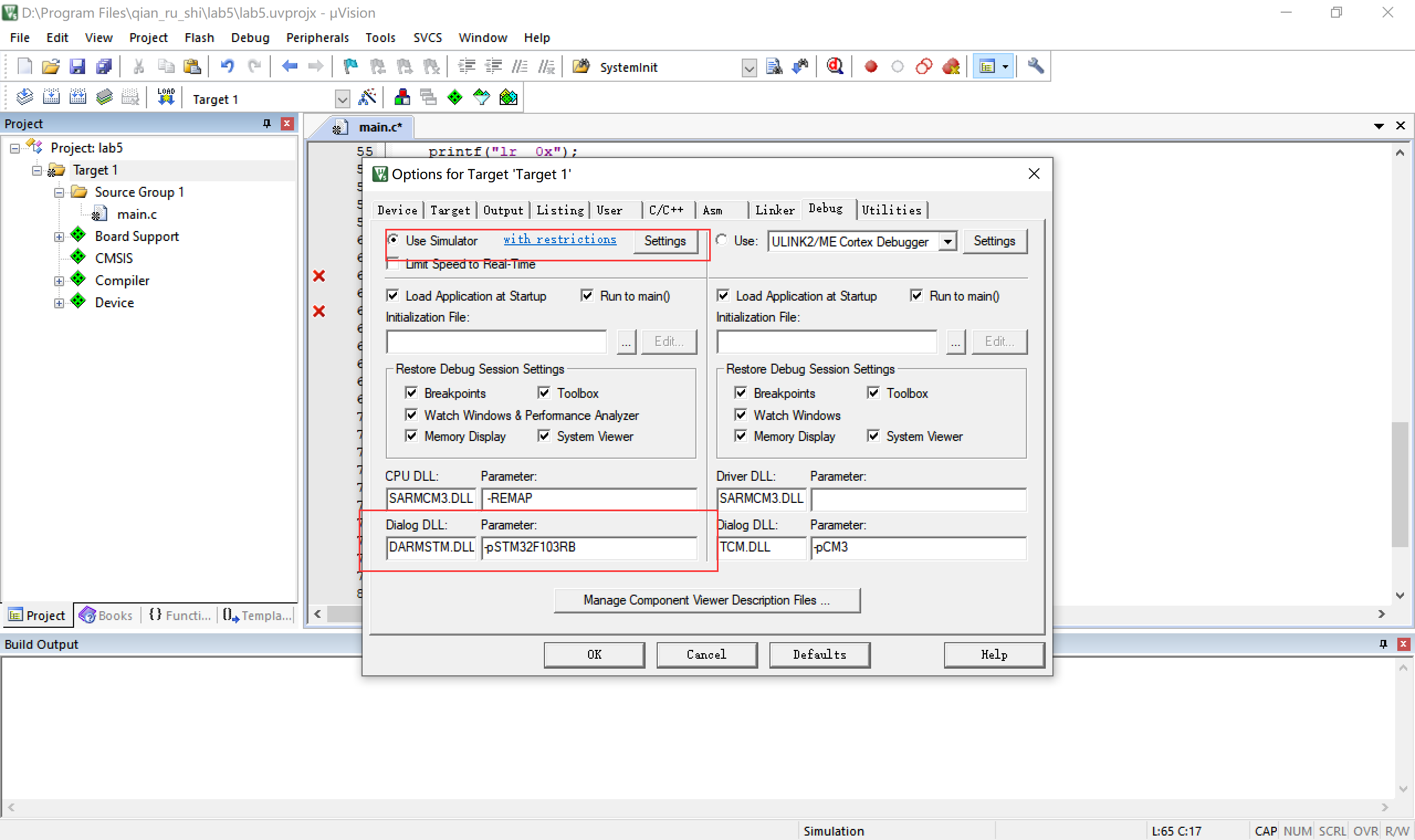
（4）异常处理模式使能

（5）异常处理程序开始执行。

# 实验步骤

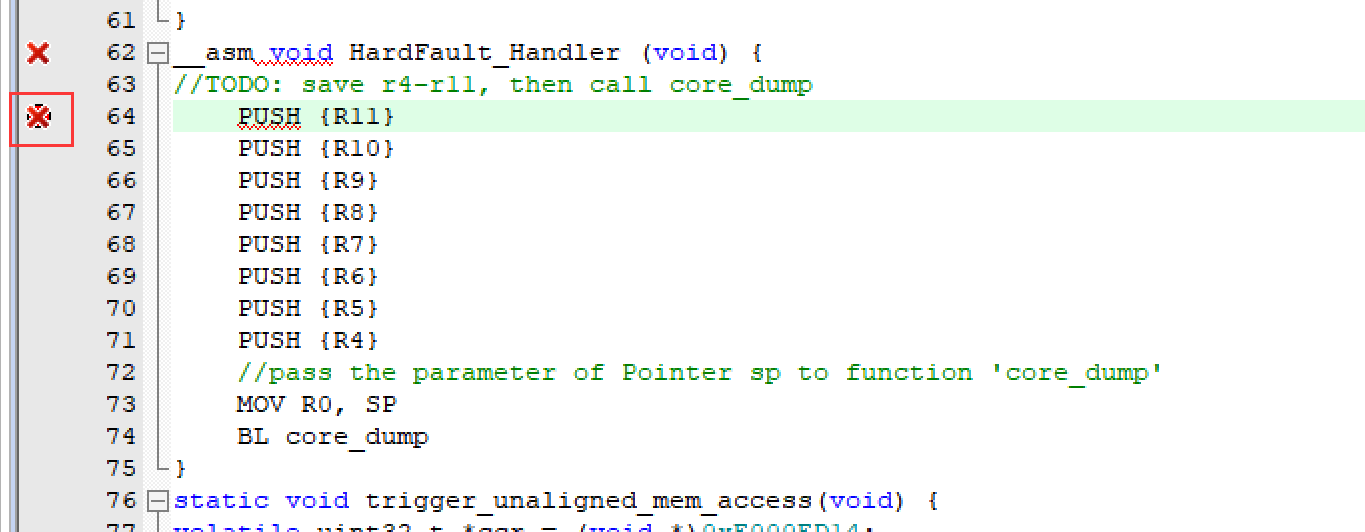
1.按照 lab2 的做法，创建新工程 lab5，并配置工程调试属性为使用模拟器调

试。

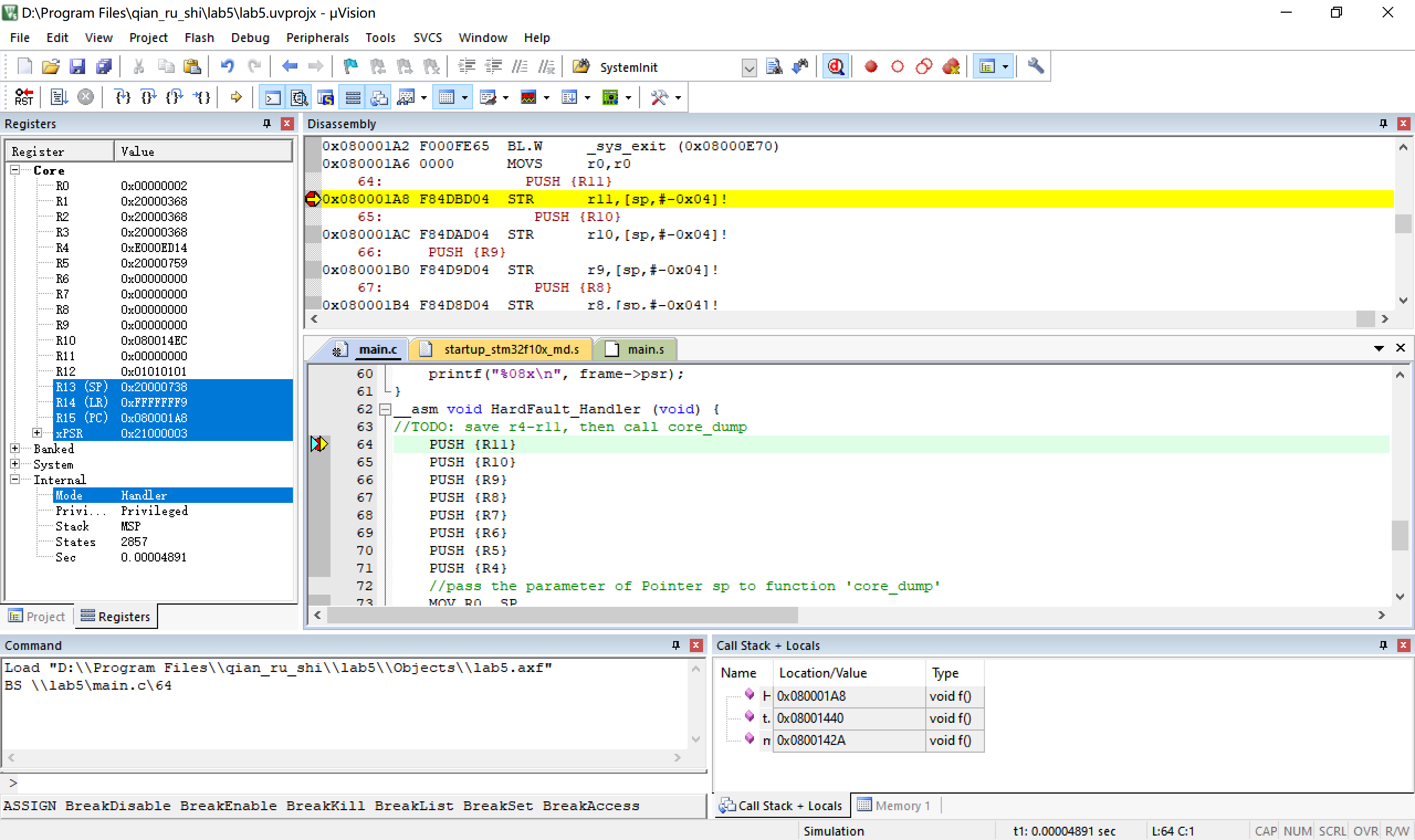


2.添加实验所给代码，并完善要填写的部分，然后在HardFault\_Handler函数的第一行加上断点：

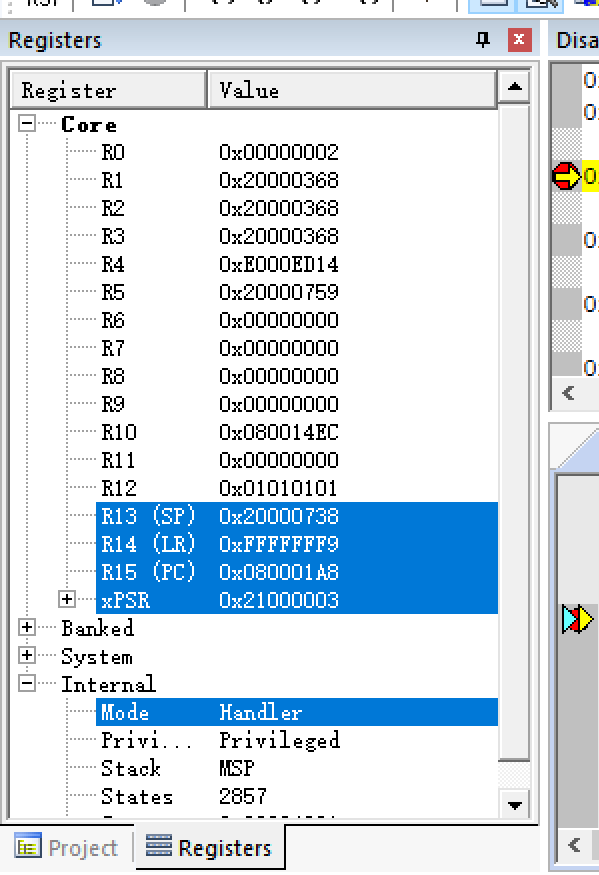




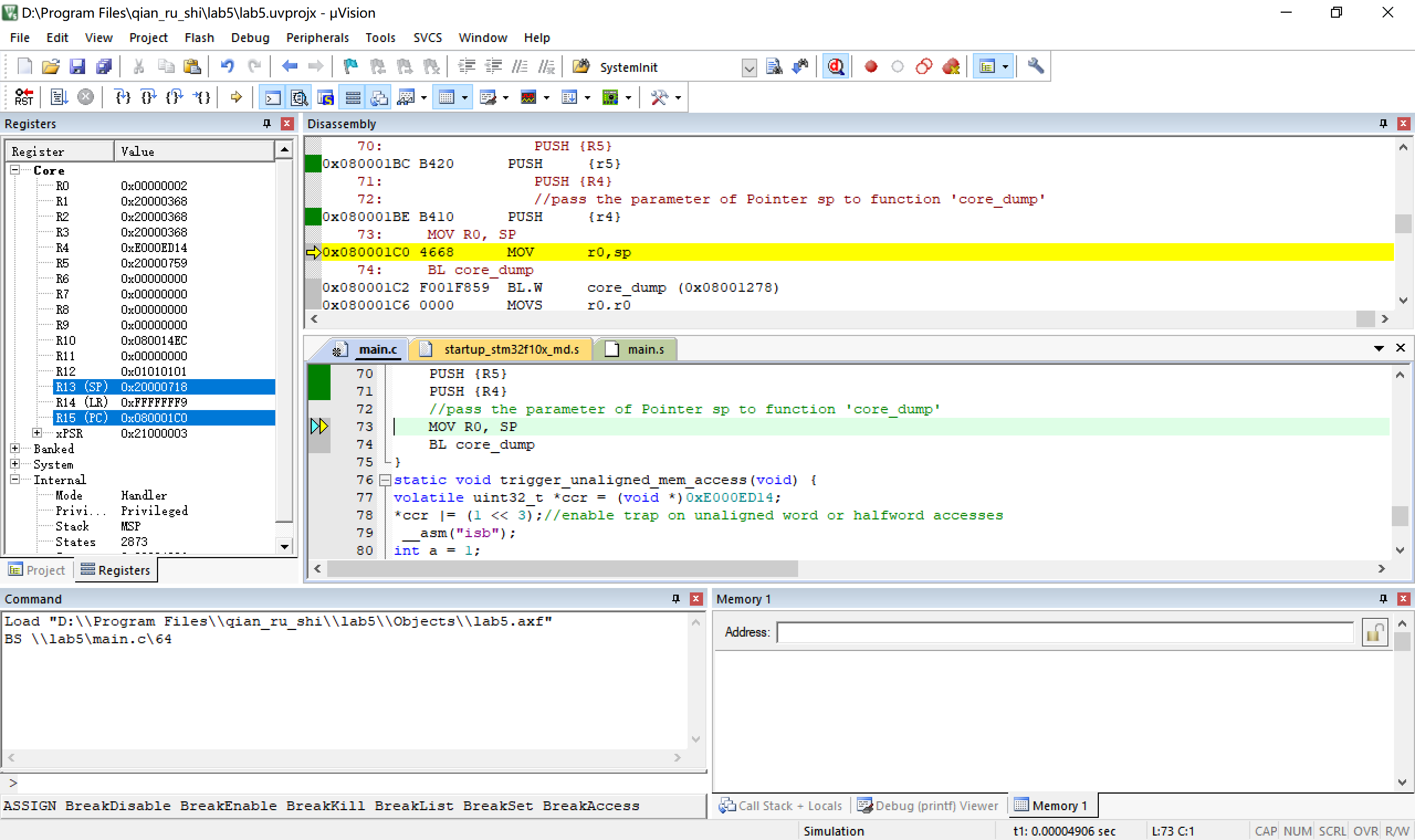
3.开始调试，执行到HardFault\_Handler函数第一行代码处：



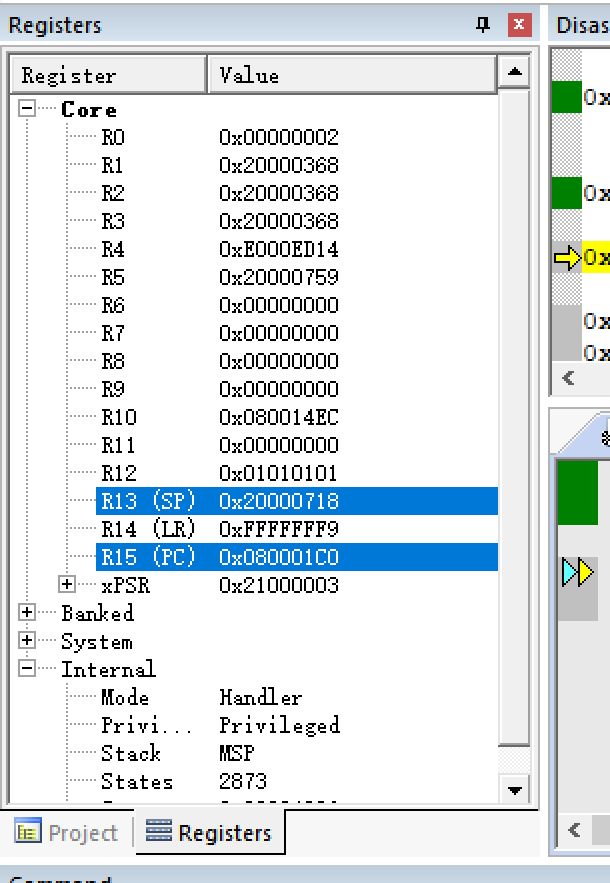
可以看到，栈顶位置为0x20000738，栈中从下到上分别保存了PSR、PC、LR、R12、R3、R2、R1、R0。



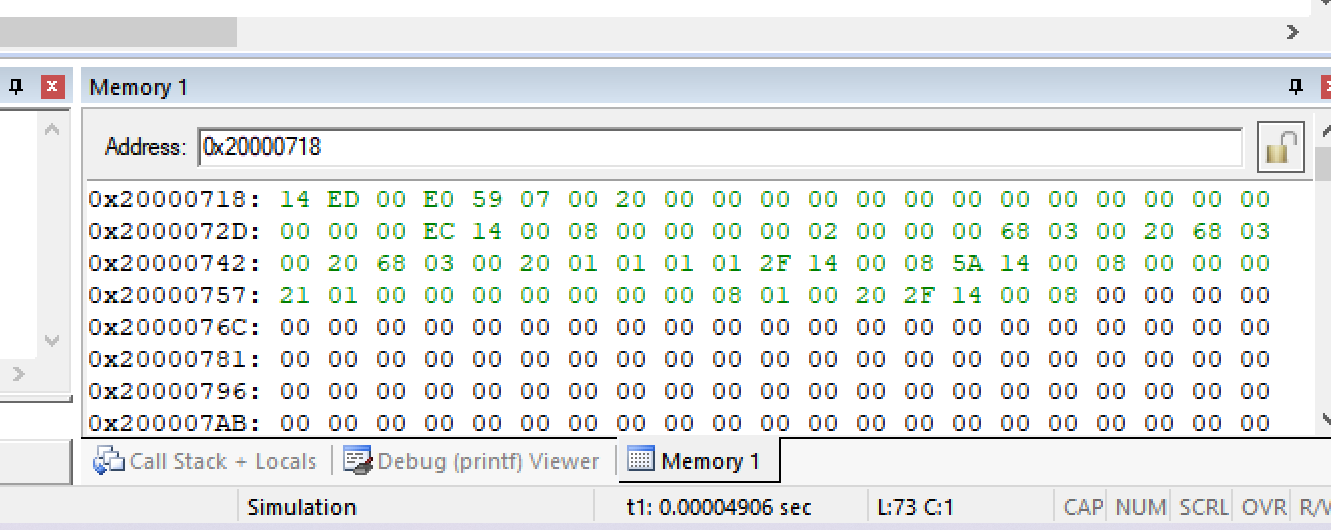
单步执行，直到R4-R11全部进栈：



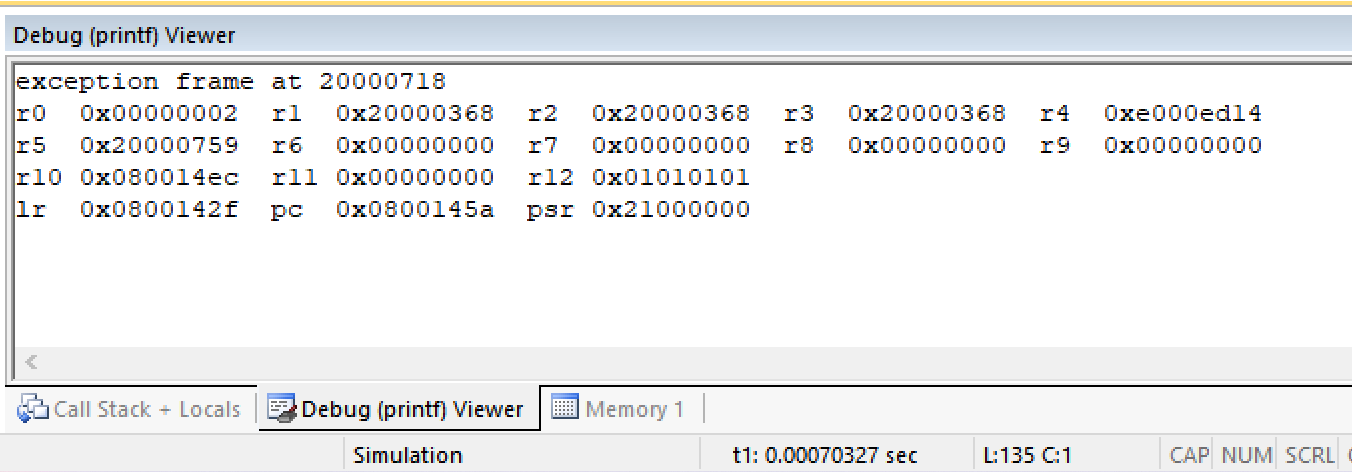
可以看到，栈顶位置从0x20000738变成了0x20000718



在查看Memory窗口输入栈顶指针位置，可以发现和上图一致：

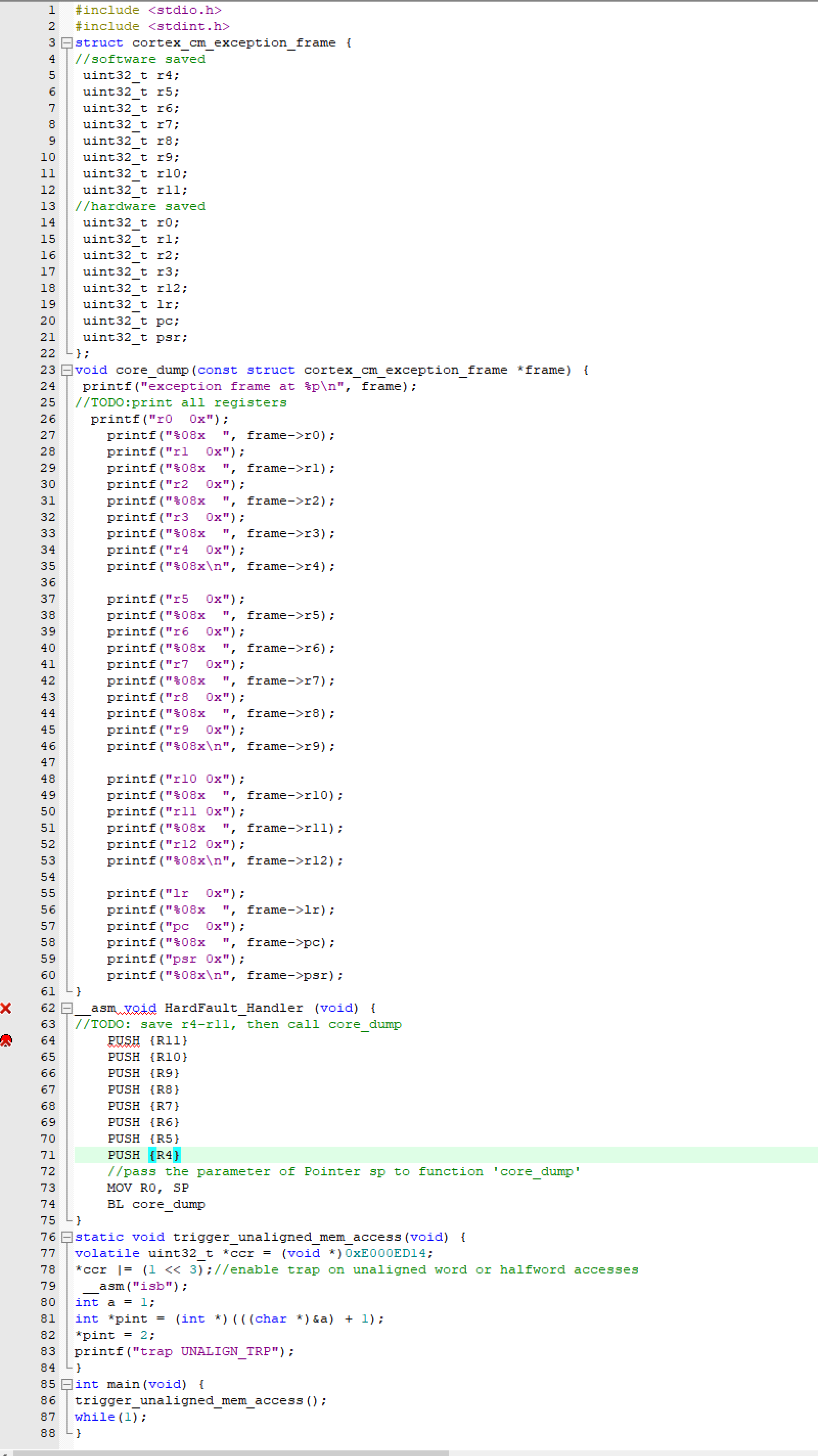


继续运行，可以看到程序输出结果与上述是一致的，所以运行成功：



# 实验方案与实现

## 源代码



# 实验结果与分析

在实验步骤中呈现了。

# 实验总结

通过本次实验，我学习理解了Cortex-M 处理器在发生异常时，进入异常处理程序的过程。实验中一开始不知道如何下手，查阅资料以及与同学讨论后有了思路，解决了问题，收获颇多。