《嵌入式系统程序设计实践》

实验报告

|  |  |
| --- | --- |
| 实验名称： | 汇编指令调试实验 |
| 姓 名： |  |
| 学 号： |  |

电子信息与通信工程学科专业实验教学中心

**实验名称：** 汇编指令调试实验

**实验环境：** keil

**实验目的：**

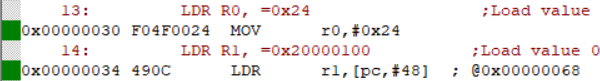
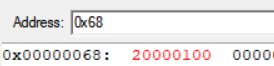
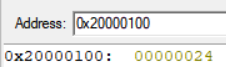
1. 探究汇编指令的表达、调试。
2. 分析汇编指令执行后存储器和寄存器中的数据与xPSR、sec等的变化。

**实验要求：**

1. 进行实验任务 1 或 2。
2. 比较汇编指令和反汇编指令的区别并分析。
3. 调试时注意观察寄存器、存储器、xPSR、Sec 的变化并分析。

**实验内容及分析：**

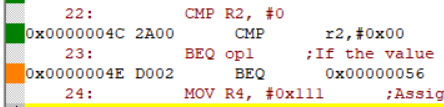
一、以下为starup.s 各条指令的简单注释理解：

1. EXPORT Reset\_Handler
2. AREA RESET, DATA, READONLY
3. EXPORT \_\_Vectors
4. \_\_Vectors
5. DCD 0x20020000
6. DCD Reset\_Handler
7. space 36
8. DCD dummy
9. AREA |.text|, CODE, READONLY
10. ENTRY
11. Reset\_Handler   PROC        ;The above code is startup code.
13. LDR R0, =0x24           ;Load value 0x24 to r0.
14. LDR R1, =0x20000100     ;Load value 0x20000100 to r1.
15. STR R0, [R1]            ;Store the value in r0 to the memory whose address is pointed to by the value in r1.
16. MOV R2, #0x123          ;Assign 0x123 to r2.
17. ADD R3, R1, #0x15       ;Add 0x15 to r1, then put the result into r3.
18. push {R0,R1}            ;push the value in r0 and r1 into the stack.
19. MOV R2, #0              ;Assign 0 to r2.
20. MOV R3, #0              ;Assign 0 to r3.
21. pop {R2,R3}             ;Pop 2 values inside the stack to r2 and r3.
22. CMP R2, #0              ;Compare the value in r2 with 0.
23. BEQ op1        ;If the value in r2 equals to 0, jump to op1. It check the flag bit in xPSR like N, Z, C, V to judge.
24. MOV R4, #0x111          ;Assign 0x111 to r4.
25. B op2                   ;Jump to op2 unconditionally.
26. op1                         ;Identifier op1.
27. MOV R4, R3              ;Assign the value in r3 to r4. But this instruction doesn't execute.
28. op2                         ;Identifier op2.
29. svc 0                   ;SVC raise exception.Mode change into Handler. Push current data into stack to save.
30. MOV R5, #0x1234         ;Assign 0x1234 to r5.
31. MOV R6, #0x6789         ;Assign 0x6789 to r6.
32. B .                     ;Jump to current unconditionally.Infinite loop.
33. ENDP
34. dummy PROC                  ;Exception handler.
35. BX LR                   ;Jump to the address pointed by LR register i.e.r14 .
36. ENDP
38. NOP
39. END
40. 针对几个重点做更详细的解释
41. 每一条指令都有自己的大小，不同的指令会有不同的大小，所以调试时PC有时加2，有时加4。
42. 这里同样是LDR指令，但其反汇编指令是不同的。当LDR的第二个参数前面是“=”时，LDR是伪指令。
43. 当立即数长度没超过8位时，为优化时间，LDR被反汇编成MOV指令；
44. 当立即数长度超过8位时，超过了MOV反汇编指令对立即数的8位限制，LDR伪指令是可以读取大范围地址或者访问内存的，在这里是通过PC偏移地址在0x68 DCW分配存储单元，将0x20000100赋给它，再通过PC偏移地址读取一个32位的立即数赋给r1（也就是文本池访问）。（LDR补充了MOV汇编指令不能访问内存的缺陷）。
45. 因为两条指令执行的反汇编指令是不同的，所以运行时间Sec也不同，第一条是8，第二条是17。
46. 偏移寻址[Rn,#offset]，带有[ ]的都是偏移寻址，寻找的地址是“Rn的值+#offset”。这里偏移是0，所以将R0的值存储在R1的值所指的存储器中。
47. ①、首先要理清楚，在我们写汇编的角度，我们是可以只用MOV来进行数据传输和立即数加载，而在仿真的过程中可以发现MOV反汇编指令包括了MOV、MOVW、MOVT，MOVL。寄存器间数据传输时汇编和反汇编指令都是MOV，而在立即数加载时，当我们写MOV汇编指令时，编译器就会根据立即数长度来自动选择对应的反汇编指令，当然我们也可以直接指定某个指令来加载立即数。
48. 接下来是在立即数加载中他们之间的区别（反汇编指令）：MOV是16位指令，加载

长度不超过8位的立即数到目的寄存器中；MOVW是32位指令，加载长度不超过16位立即数 到目 的寄存器的低16位中；MOVT是 32位指 令，加载长度不超过16位立即数到目的寄存器的高16位中 。

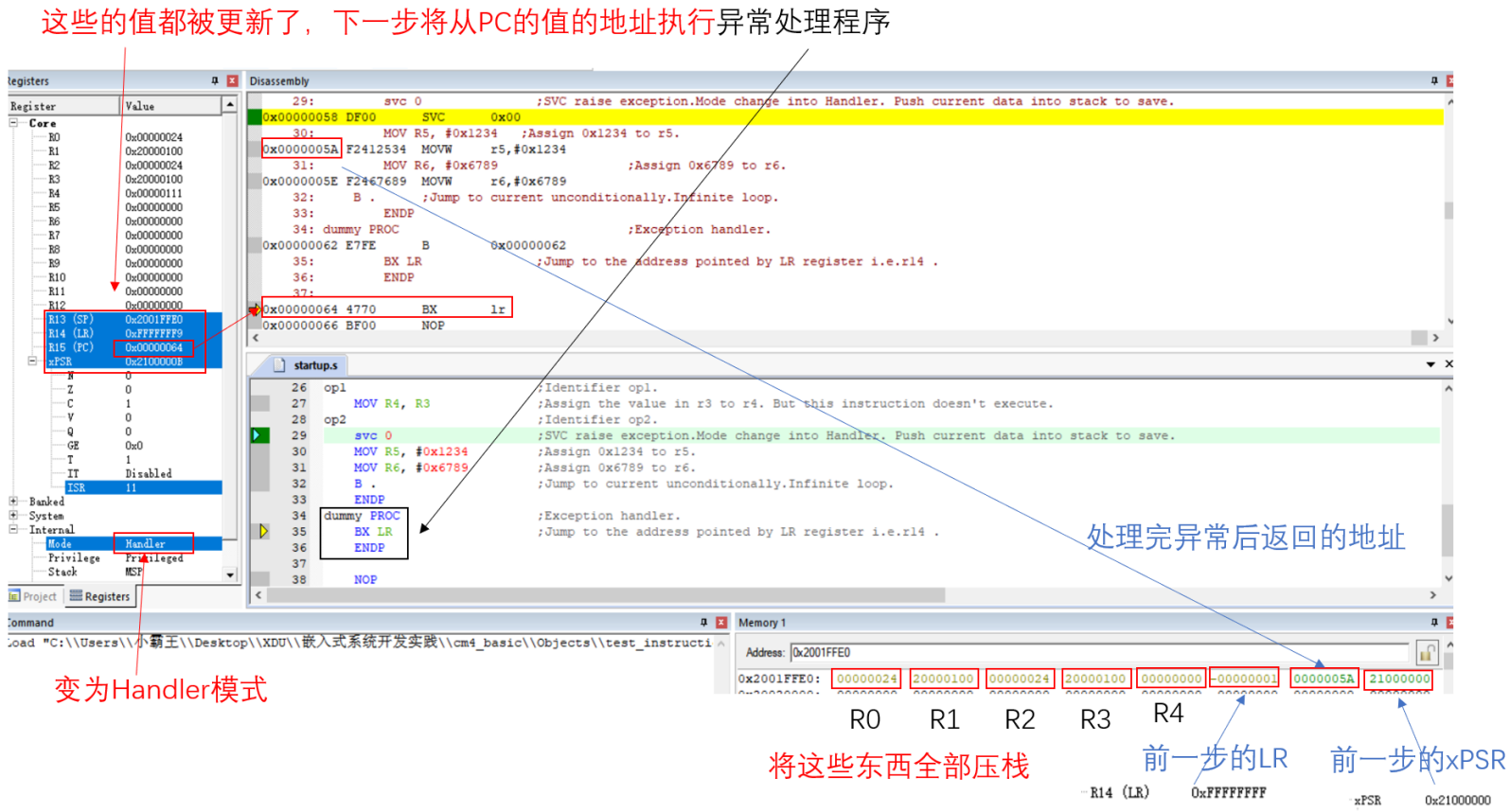
1. 栈（stack）存储。保存数据称为压栈（push），读取数据称为弹栈（pop）。下面都是针对Cortex M4中的栈进行描述：
2. 栈底（指针）是保持不动的。
3. 空栈是栈顶和栈底相同重合。满栈是栈顶达到栈空间上限。
4. Push时栈顶先上移（栈顶指针先减小），再在栈顶存入数据。Pop时先读取栈顶数据，栈顶再下移（栈顶指针再增加）。先压R1（0x20000100），再压R0（0x24）。先弹0x24到R2，再弹0x20000100到R3。对于只弹出一个数据则如下，弹0x24到R2此时的栈顶指针SP为。
5. 比较两数CMP。两比较数相减（前减后），通过更新后的xPSR中的标志位来判断他们之间的大小关系。
6. B跳转。

①、有条件跳转：BEQ（相等时跳转）、 BNE(不等时跳转).........

这里是比较了R2的值和0的大小，如果两者相等，则跳转到标记符op1指的地址（0x56）来执行后续代码，可以看到这里不符合条件（不成立则显示橙色），不跳转。

②、无条件跳转：B 这里是无条件跳转到op2指的地址（0x58）执行后续代码。跳转到当前：“B .” 相当于死循环。

③、状态跳转BX（Branch and exchange instruction set）：BX Rm，跳转到Rm数据的地址，若Rm的数据的第0位是1，则将XPSR中的T标志位置为1（执行Thumb指令集）；若Rm的数据的第0位是0，则将XPSR中的T标志位置为0（执行arm指令集）。 在这里处理完异常后，跳转到异常中断时存储在连接寄存器LR中的地址二进制是-111，第0位是1，T置为1。

7.SVC 0。 自己引起的异常中断，工作模式变为Handler（Handler模式可以获得各种高级权限，而Thread模式只能拥有一般权限），流程如图

**实验总结**

本次实验认真探究了starup.s中每一句汇编指令的实现流程和实现效果，重点分析总结了LDR、偏移寻址、STR、MOV、ADD、stack、CMP、B、SVC的用法和原理，在搜索资料学习的基础上把个人的疑惑点和重点知识进行了解释。后续的学习也要多看官方文档。