计算机组成原理

实验1 RISC-V汇编程序设计

2025 · 春

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

实验课程介绍

- 理解计算机组成原理
- 掌握CPU主要部件的设计方法
- 计算机考研课
- 计算机体系结构方向的基础
- 在线指导书：
  - ✓ https://organ.p.cs-lab.top/

实验课程介绍

- 实验学时：**16**学时
- 实验总分：**30**分

	实验项目	学时	分数	实验平台
实验1	RISC-V汇编程序设计	4	6	RV32 GNU、RARS
实验2	浮点运算器设计	4	8	Vivado 2018.3
实验3	高速缓存器设计	4	8	Vivado 2018.3
实验4	AXI总线接口设计	4	8	Vivado 2018.3

实验目的

- 了解C语言**从编译到汇编**的过程，了解C程序和汇编语言的对应关系
- 熟悉**子程序工作流程**，理解子程序工作原理
- 熟悉RISC-V汇编程序语法，掌握基本**汇编程序的编写**
- 掌握RISC-V汇编工具RARS的使用方法

实验内容

- 1、用C语言编写程序，实现IEEE754加法（程序已提供）
- 2、在RV32汇编环境中，完成C程序的预编译、编译、汇编、链接及执行的过程

实验内容

- 3、编写RV32汇编程序，实现IEEE754加法，要求：
  - 源数据均存放在数据段，当成32位无符号数读取

```
1 .data
2     float1: .float -9.777     # 0xc11c6e98
3     float2: .float -1.234     # 0xbf9df3b6
4
5 .text
6     ......
```

  - 规定：均为规格化数、符号相同、float1阶码不小于float2阶码
  - **正确使用**子程序、不可使用伪指令

实验原理

◆ **编译** 【高级语言(.c) -> 汇编语言(.s/.asm)】

分两步：预编译/预处理 + 编译

A. **预编译** (gcc.gnu.org/onlinedocs/cpp/)

- 去除注释 —— 单行注释//、多行注释/* */
- 处理预编译指令
  - 包含文件 —— 将被包含的文件插入到相应的#include语句处
  - 替换宏定义 —— #define常量替换、表达式代入
  - 处理条件预编译指令 —— #if、#ifdef、#elif、#else、#endif

B. **编译** ("*Compilers: Principles, Techniques, & Tools*")

对预处理后的源文件进行词法分析、语法分析、语义分析、中间代码生成、代码优化以及目标代码生成，得到汇编代码

实验原理

◆ **汇编** 【汇编语言(.s/.asm) -> 目标文件/对象文件(.o)】

将汇编代码翻译成CPU能识别的机器码，并按照ELF文件的格式标准，将机器码存储在目标文件中

- ELF (Executable & Linkable Format)：一种目标文件/链接库文件/可执行文件的标准文件格式 (flint.cs.yale.edu/cs422/doc/ELF_Format.pdf)
- 目标文件中存储的是机器码 (Machine Language)

每一个汇编语句几乎都对应一条CPU指令。因此，汇编比编译简单，汇编器只需根据汇编语句和CPU指令的对照表一一翻译即可

- “几乎” —— 伪指令等价于多条指令，如RISC-V的li指令可能等价于lui和addi)
- 汇编语句与CPU指令的对照表：ISA手册 (riscv.org/technical/specifications/)

实验原理

◆ 链接 【目标文件(.o) -> 可执行文件】

根据重定向表，将各个目标文件及库文件链接在一起，成为可执行文件

- Link：把各.o文件中的代码段 (text segment)、数据段 (data segment) 等全部“拼”在一起
- 重定向表由编译器生成，存放在含有主函数的目标文件中，记录了哪些符号以何种方式 (绝对地址/相对地址) 重定位到哪里
  - 查看重定向表的命令：gcc -r main.o
  - Ref：mp.ofweek.com/ee/a656714328207
- 库文件的链接方式：静态链接、动态链接
  - 静态链接：把目标文件和库文件直接链接在一起
  - 动态链接：把目标文件和库文件的描述信息链接起来，运行时加载库代码

实验原理

◆ 加载执行

一般由OS加载可执行文件

- 在内存中创建代码段、数据段、堆栈段地址空间
- 拷贝用户参数，初始化寄存器
- 跳转到用户程序，设置PC

◆ 反汇编 【机器码 -> 汇编程序】

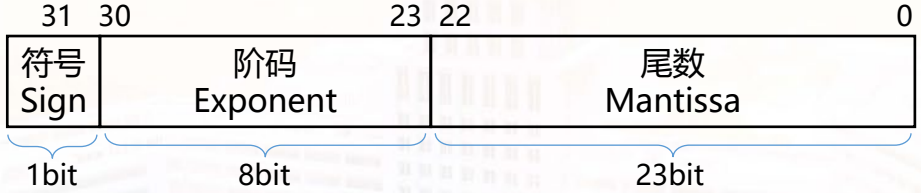
反汇编是用于调试和定位处理器问题时最常用的手段之一

- ELF文件并非文本文件，无法用编辑器打开并查看代码
- 若想查看ELF文件中的指令和数据，需要借助反汇编

实验原理

◆ IEEE754单精度浮点数

- 规格化数的组成：符号S、阶码E、尾数M’ = {1, M}



- 假设$E_x \geq E_y$，则求解z = x + y的步骤为：
  - ① 求阶差：$\Delta E = E_x - E_y$
  - ② 对阶：$M_y' >>= \Delta E$
  - ③ 尾数运算：$M_z' = M_x' + M_y'$
  - ④ 规格化：若加法溢出，即$M_z'[24]$不为0，则右规：$M_z' >>=1$，$E_z = E_x + 1$
    否则$E_z = E_x$，最终得$z = \{S_z, E_z, M_z'[22:0]\}$

实验环境

虚拟机：WSL

操作系统：Debian 12

编译、汇编和链接：RISC-V 32bit GNU

RISC-V模拟器：Spike仿真器 + riscv-pk代理内核（执行RISC-V程序）

自行构建方案:

organ.p.cs-lab.top/lab1/A-envir/

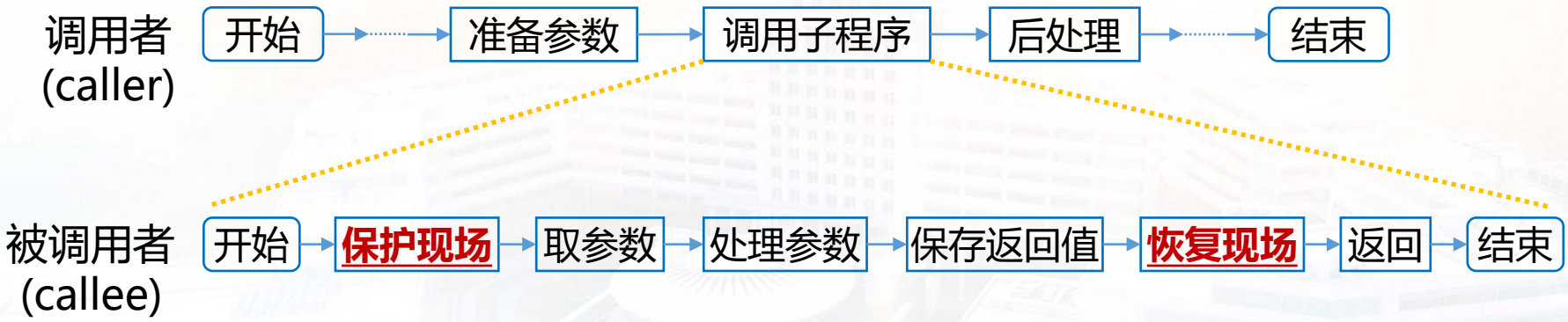
实验演示

以IEEE754加法的C程序为例

演示编译、汇编等过程

实验原理

◆ **子程序**执行流程：



实验原理

◆ **子程序**框架：

```
14  .text
15  MAIN:
16      ......
17      jal  ra, SOME_FUNC        # Call a sub-routine.
18      ......                    # Here is where the sub-routine returns.
19      ori  a7, zero, 10         # Set system call number(10 for termination).
20      ecall                     # This program terminates here.
21
22  SOME_FUNC:
23      push s0
24      push s1
25      ......                    # Assume that s0 and s1 are modified here.
26      pop  s1
27      pop  s0
28      jalr zero, 0(ra)          # The sub-routine returns.
```

- 关键点：调用、返回、**保护现场**、**恢复现场**

实验原理

◆ 保护现场：

- RISC-V规定sp、s0~s11需要callee保护，a0~a7、t0~t6需caller保护
- 若子程序内部修改sp、s0~s11的值，则需在保护现场时备份寄存器
- 方法：使用push、pop宏定义保护现场、恢复现场

```
.macro push %a
    addi sp, sp, -4
    sw   %a, 0(sp)
.end_macro

.macro pop %a
    lw   %a, 0(sp)
    addi sp, sp, 4
.end_macro
```

```
SOME_FUNC:
    push t0
    push t1
    ......
    pop  t1
    pop  t0
    jalr zero, 0(ra)
```

实验原理

- **伪指令**区分：
  - 查指令手册
  - 借助RARS判断：
    - “Basic”和“Source”下显示的指令不同则是伪指令

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00000000	0x000022b7	lui x5,2	16: lui t0, 0x00002
	0x00000004	0x000022b7	lui x5,2	17: li t0, 0x00002000
	0x00000008	0x00028293	addi x5,x5,0	
	0x0000000c	0x0002a503	lw x10,0(x5)	18: lw a0, 0(t0)
	0x00000010	0x010000ef	jal x1,0x00000010	19: jal ra, FUNC
	0x00000014	0x00c000ef	jal x1,0x0000000c	20: jal FUNC

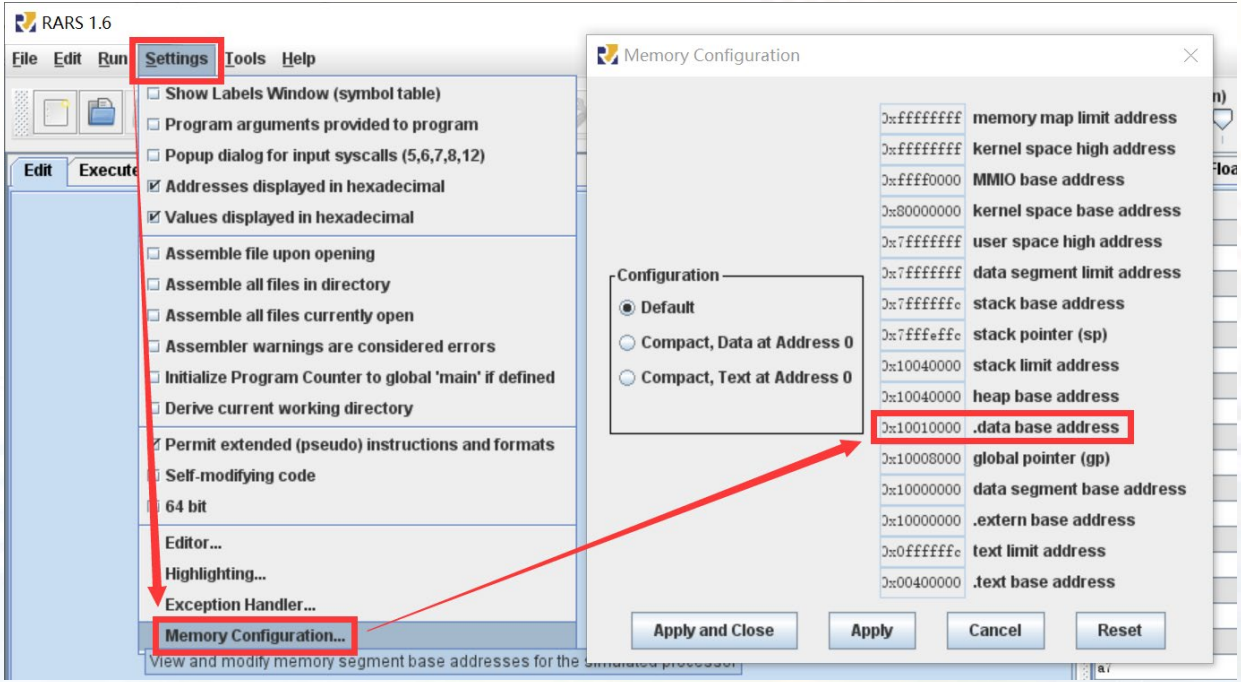
实验步骤

1. 完成C程序到汇编程序，再到机器码的过程
   - ① 编写C程序
   - ② 预编译，得到预编译文件 (.i)
   - ③ 编译，得到汇编文件 (.s)，查看汇编代码
   - ④ 汇编，得到目标文件 (.o)，反汇编并查看目标文件的机器码
   - ⑤ 生成可执行文件，反汇编并查看可执行文件的机器码

实验步骤

2. 编写RV32汇编程序实现浮点加法功能，并在RARS上调试、运行

① 打开RARS，查看存储配置模式



数据段基地址：
0x10010000

实验步骤

2. 编写RV32汇编程序实现浮点加法功能，并在RARS上调试、运行

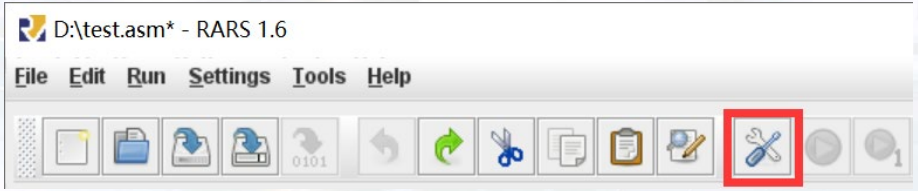
② 编写RV32汇编程序

```
1 .data
2     float1: .float -9.777     # 0xc11c6e98
3     float2: .float -1.234     # 0xbf9df3b6
4
5 .text
6     ......
```

地址为0x10010000

地址为0x10010004

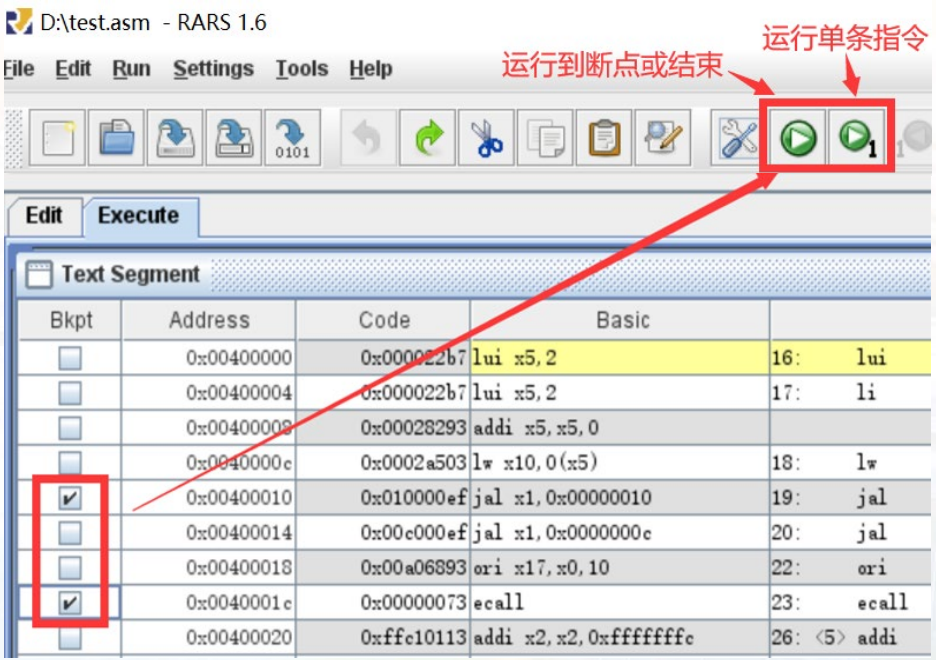
③ 对源代码进行汇编



实验步骤

2. 编写RV32汇编程序实现字符串匹配功能，并在RARS上调试、运行

④ 调试和运行



验收与提交

- **验收内容**
  - 课上检查浮点加法的汇编程序能否正确运行：1.2分
- **提交内容**
  - 预编译文件(.i)、汇编文件(.s)、目标文件(.o)及其反汇编文件、可执行文件及其反汇编文件：1.2分
  - 浮点加法汇编程序（fp_add.asm）：1.2分
  - 实验报告（按模板完成）：2.4分
- 将上述文件打包成.zip，以“学号_姓名.zip”命名提交到作业系统
  - 注意：**如有雷同，双方均0分！**

开始实验