计算机组成原理 实验1 从C语言到机器码



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
- 考研课
- 计算机体系结构方向的基础
- 在线指导书:
 - √ https://hitsz-cslab.gitee.io/organ/



实验课程介绍

• 实验学时: 12学时

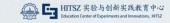
• 实验总分: 20分

序号	实验项目	学时	分数	实验平台
实验1	从C语言到机器码	2	3	RISC-V GCC工具链
实验2	寻找回文子串	2	3	RARS
实验3	浮点运算器设计	4	7	Vivado
实验4	AXI-Lite总线接口设计	4	7	Vivado

实验目的

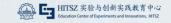
- 了解C语言到汇编语言的编译过程,熟悉RISC-V汇编语言程序
- 了解<u>汇编语言到机器码</u>的<mark>汇编</mark>过程,熟悉RISC-V汇编常用指令机器码
- 掌握原码一位乘算法





实验内容

- 1、用C语言实现原码一位乘法:
 - · 操作数为8bit原码(学号前2位★后2位),结果为16bit
 - 打印运算的结果
 - ◆ 要求: 实现正整数乘法,不需处理符号位; 且不可使用乘号
 - · 例: 学号是200110618的同学, 输出的结果应为360
- 2、在RISC-V汇编环境中,完成C程序的编译、汇编、链接、加载执行过程,填写实验报告



实验原理 - 原码一位乘

	_	积	乘数 丢弃	说明
***		00000000	1101	高部分积置 0 , $\mathbf{Z_0} = 0$
X = 1110	+	1110	HUNU	乘数为 1 , 加 X (* 2 ⁿ)
Y = 1101		$\underline{1\ 1\ 1\ 0\ 0\ 0\ 0} \underline{0}$		逻辑右移→1,
求 $[X \times Y]_{\mathbb{R}}$		01110000	<u>0 1 1 0 <i>1</i></u>	得 Z ₁
	+	0000		乘数为 0 , 加 0(*2n)
		01110000		逻辑右移→1,
		$0 \underline{0} \underline{1} \underline{1} \underline{1} \underline{0} \underline{0} \underline{0}$	0 0 <u>1</u> 1 <u>0</u> 1	得 Z 2
		1110		乘数为 1, 加 X (*2 ⁿ)
		100011000		逻辑右移→1,
		$0 \underline{10001100}$	00011101	得 Z ₃
		1110		乘数为 1, 加 X (*2 ⁿ)
	_	$\frac{101101100}{1101100}$	0.000110	逻辑右移→1,
		010110110	0 0 0 0 110 1	得 Z ,结束 导HTSZ 实验与创新实践

实验原理 - 原码一位乘

优化版本

部分积(乘数暂存) 丢弃	说明
00001101.	高部分积置 0 , $Z_0 = 0$
+ 1110	乘数为 1, 加 X (*2 ⁿ)
1110 1101	逻辑右移→1,
0 1 1 1 0 1 1 0 7	得 Z ₁
+ 0000	乘数为 0, 加 0(*2n)
<u>0111 0110 </u>	逻辑右移→1,
0 0 1 1 1 0 1 1 91	得 Z 2
+ 1110	乘数为 1, 加 X (*2 ⁿ)
10001 1011	逻辑右移→1,
0 1 0 0 0 1 1 0 1 101	得 Z_3
+ 1110	乘数为 1,加 X (* 2 ⁿ)
10110 1101	逻辑右移→1,
0 1 0 1 1 0 1 1 0 1 10	得Z ₄ ,结束

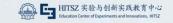
◆ 编译 【高级语言(.c) -> 汇编语言(.s/.asm)】

分两步: 预编译/预处理 + 编译

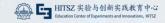
A. 预编译 (gcc.gnu.org/onlinedocs/cpp/)

- 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<u>词法分析</u>、<u>语法分析</u>、<u>语义分析</u>、<u>中间代码生成</u>、 <u>代码优化以及目标代码生成</u>,得到汇编代码



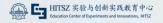
- ◆ 汇编 【汇编语言(.s/.asm) -> 目标文件/对象文件(.o)】 将汇编代码翻译成CPU能识别的机器码,并按照ELF文件的格式标准,将机器码 存储在目标文件中
 - ELF (Executable & Linkable Format): 一种目标文件/链接库文件/可执行文件的标准文件格式 (flint.cs.yale.edu/cs422/doc/ELF_Format.pdf)
 - 目标文件中存储的是<u>机器码</u> (Machine Language)
 - 每一个汇编语句几乎都对应一条CPU指令。因此,汇编比编译简单,汇编器只需根据汇编语句和CPU指令的对照表——翻译即可
 - "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/)



◆ 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<u>哪些符号</u> 以<u>何种方式(绝对地址/相对地址)</u>重定位<u>到哪里</u>
 - 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

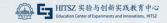


◆ 加载执行

- 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



实验环境

虚拟机: Oracle VM VirtualBox 7.0.4

操作系统: Ubuntu 22.04.2

编译、汇编和链接: riscv64 GCC

riscv 模拟器: spike (执行 riscv 程序)

自行构建方案:

hitsz-cslab.gitee.io/organ/lab1/envir/ (21级满洋同学贡献)

基于Ubuntu的riscv_v tools安装指南 (gitee实验包)

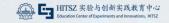
实验扩展: 掌握上述工具链的安装和使用



实验演示

以Hello World为例

演示编译、汇编等过程



实验步骤

- 1. 编写C语言程序
- 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),查看机器码
- 5. 生成可执行文件
- 6. 反汇编,查看反汇编文件
- 7. 根据实验过程,按照模板完成实验报告

注意:在汇编生成.o文件时,可添加-march=rv64g的选项,使汇编器只为主函

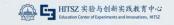
数生成32位机器码

实验提交

• 提交内容

- · C程序、汇编程序、机器码 (.o文件及可执行文件)、反汇编文件: 1分
- 实验报告(按模板完成):2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!





1920

III 大

